

# A Semi Distributed Approach for Min-Max Fair Car-Parking Slot Assignment Problem

E. Alfonsetti, P. C. Weeraddana, *Member, IEEE*, and C. Fischione, *Member, IEEE*

**Abstract**—Designing efficient car parking mechanisms that can be potentially integrated into future intelligent transportation systems is of crucial importance. Usually, the related design problems are combinatorial and the worst-case complexity of optimal solution approaches grows exponentially with the problem sizes. Therefore, such optimal approaches are not scalable and practically undesirable. As a result, almost all existing methods for parking slot assignment are simple and greedy approaches, where each car is assigned a free parking slot, which is closer to its destination. Moreover, no emphasis is placed to optimize the social benefit of the users during the parking slot assignment. In this paper, the fairness as a metric for modeling the aggregate social benefit of the users is considered and a distributed algorithm based on Lagrange duality theory is developed. The proposed algorithm is gracefully scalable compared to the optimal methods. In addition, it is shown that the proposed car parking mechanism preserves privacy in the sense that any car involved in the algorithm will not be able to discover the destination of any other car during the algorithm iterations. Numerical results illustrate the performance of the proposed algorithm compared to the optimal assignment and a greedy method. They show that our algorithm yields a good tradeoff between the implementation-level simplicity and the performance. Even though the main emphasis in this paper resides in the car parking slot assignment problem, our formulation and the algorithms, in general, can also be applied or adopted in fair agent-target assignment problems in other application domains.

**Index Terms**— Intelligent transportation systems, optimization methods, algorithms, privacy

## I. INTRODUCTION

The car is certainly one of the most used means of transport, but its introduction, despite having brought comfort and simplification of life, has generated well-known problems of increasing traffic, and thus clogging roads in the city centers. Especially in large cities, these problems are pronounced by hundreds or even thousands of drivers who are looking for parking slots during their daily activities. In [1] it is claimed that seeking for parking slots (*cruising*) can account for more than 10% of the local circulations in central areas of large cities. In [2], it is reported that cruising for open parking spaces accounts for 30% of the traffic, causing undesired congestion in big cities. In addition, cruising creates additional delays and drivers can even spend up to 10-20 minutes before they could find a proper parking slot. According to a recent British study, it is estimated that a person who owns a car

in big cities can take an average of 6 to 20 minutes to search for an empty parking slot, which accounts for monetary losses (e.g., deterioration, unnecessary fuel wastage), as well as for nonmonetary expenses (e.g., frustration, psychophysical stresses).

In the context of future intelligent transportation systems (ITS), there are many relevant research activities, which design various traffic congestion control mechanisms, see [3]–[7] and references therein. However, [2] suggests that designing efficient car parking mechanisms, which are instrumental in directly reducing the cruising traffic is just as important as other related methodologies to minimize undesired traffic conditions, especially, in big cities.

Several research attempts have been made in the field of ITS, which support drivers to locate a free parking slots, see [8]–[17]. In general, these existing methods employ a central authority (CA) who is responsible for providing the underlying infrastructure. For example, thousands of sensors have to be deployed to detect the availability of free parking slots [8]–[14], [17]. The authors in [15], [16] have considered a stage of vehicular ad hoc networks, where advanced roadside units are widely deployed and every vehicle is equipped with sophisticated onboard units. In addition, efficient database management systems together with real-time communication protocols have to be implemented to meet the demands.

The proliferation of smartphones has encouraged a number of private, as well as state companies to deploy real-time car parking mechanisms in central areas of large cities, see ParkingCarma [18], Steeteline Parker [19], VehicleSense Street Parking Information Network (SPIN) [20], VehicleSense SmartLot [20], and SFMTA SFpark [21]. Like other research work proposed in [8]–[17], these methods are also relying on a central authority for providing the underlying infrastructure, such as wireless sensors, database management systems, etc.

In almost all existing methods, no emphasis has been placed to optimize an aggregate social benefit of the users during the parking slot assignment. Arguably, the existing mechanisms can be interpreted as greedy methods, where each user selects the closest free parking slot to its destination. Such a greedy method can easily account for substantial imbalances among the distances between users' parking slots and their destinations, i.e., is not fair. In other words, some users can be assigned to parking slots that are very close to their destinations while others can be assigned to parking slots that are far from their destinations. For example, Figure 1 shows a case where two cars are to be assigned to two parking slots. A greedy assignment approach yields the (car, parking slot) assignment (1, 1) and (2, 2) which accounts for a total cost of 6 (i.e.,  $1 + 5$ ) and a cost imbalance of 5. On the other hand, a

Manuscript received ... This research was supported by EU projects Hycon2, Hydrobionets, and VR project In network Optimization.

E. Alfonsetti with TerraSwarm Lab, Electrical Engineering Department, UC Berkeley, California, USA (e-mail: e.alfonsetti@berkeley.edu).

P. C. Weeraddana and C. Fischione with Electrical Engineering, KTH Royal Institute of Technology, Stockholm, Sweden (e-mail: chatw@kth.se, carlofi@kth.se).

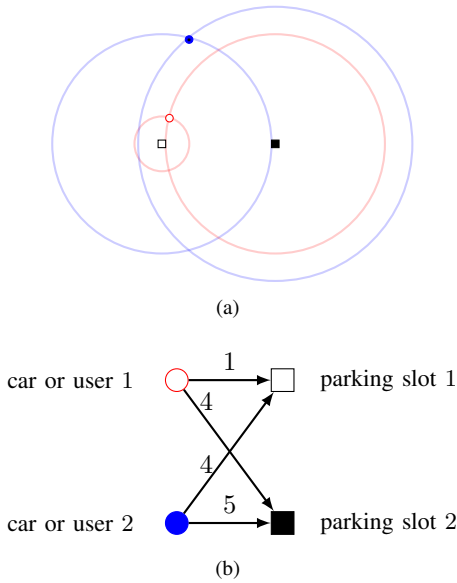


Fig. 1: Car-Parking slot assignment: (a) Physical locations of free parking slots (squares) and cars' destinations (circles); (b) Graph, where the weights denote the distances between the destinations and parking slots.

fair assignment yields the assignment  $(1, 2)$  and  $(2, 1)$  which accounts for a cost imbalance of 1.

Therefore, it is worth seeking efficient, as well as advanced algorithms that are capable of optimizing some aggregate social benefit for the users of the system. Because the involvement of a central authority is instrumental in coordinating the car parking mechanisms, optimization criteria can be integrated in to the parking assignment methods, where some aggregate social metric (utility) is considered during the assignment process. One such simple, yet appealing utility is users' fairness [22].

Ensuring privacy of the associated algorithms, which can be in various contexts, is also important, see [16]. Naturally, users would not like to publish information, such as their destinations to prevent a third party from predicting private traveling patterns. For example, government agencies can probe such information during investigations and business entities might be interested in exploiting such information to promote their products and services. Therefore, exposure of private information raises serious concerns of personal privacy.

The main contributions of this paper are as follows:

- 1) We consider the *min-max fairness* as a metric for modeling the aggregate social benefit of the users [22]. In particular, we consider the distance between parking slot and the destination that corresponds to every user. We refer to this distance, associated with any user, as the *parking distance*. Then we design an algorithm to minimize the maximum parking distance among all the users. The proposed algorithm is based on duality theory [23, Section 5]. Our formulation and the corresponding algorithm can be applied directly or with minor modifications in fair agent-target assignment problems in other application domains as well, and therefore is not restricted to the car-parking assignment.
- 2) We capitalize on dual decomposition techniques [24] and

the subgradient methods [25] to accomplish distributed implementation (among users) of the proposed algorithm with a little coordination of the central authority. Therefore, the proposed algorithm has rich scalability properties, which is indeed favorable in practice.

- 3) The proposed car parking mechanism is privacy preserving in the sense that any car involved in the algorithm will not be able to find out the destination of any other car during the algorithm iterations. This privacy is accomplished as a result of the inherent decomposition structure of the problem together with randomization of the step size of the subgradient method.
- 4) A number of numerical examples are provided to evaluate the performance of the algorithm. In addition, the proposed algorithm is compared with the optimal assignment method and with a greedy assignment method.

Thus, our solution approach for the car parking problem is fair, distributed, and is easily deployed with the coordination of a central entity. In addition, it has appealing privacy properties.

The rest of the paper is organized as follows. A description of the system model and the problem formulation is presented in Section II. In Section III, we provide the solution method to the car parking problem by using duality theory and subgradient method. Section IV presents our proposed algorithm for distributed car parking assignment problem. In Section V, we describe privacy properties of the algorithm. In Section VI, numerical results are provided. Lastly, Section VII concludes the paper.

## Notations

Boldface lower case and upper case letters represent vectors and matrices, respectively, and calligraphy letters represent sets. The set of real  $n$ -vectors is denoted by  $\mathbb{R}^n$  and the set of real  $m \times n$  matrices is denoted by  $\mathbb{R}^{m \times n}$ . We use parentheses to construct matrices from comma separated sub-matrices of agreed dimensions, e.g.,  $(\mathbf{A}, \mathbf{B}, \mathbf{C}) = [\mathbf{A}^T \ \mathbf{B}^T \ \mathbf{C}^T]^T$ . We denote by  $(\mathbf{A}_i)_{i=1,2,\dots,N}$  the matrix  $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_N)$ . The cardinality of a set  $\mathcal{A}$  is denoted by  $\text{card } \mathcal{A}$ .

## II. SYSTEM MODEL AND PROBLEM FORMULATION

A system consisting of  $M$  parking slots and a number of destinations is considered. We denote by  $\mathcal{M} = \{1, \dots, M\}$  the set of parking slots. Destinations can include *any* geographical location, such as shops, bars, banks, cinemas, casinos, houses, parks, hotels, among others. The parking slots and the destinations can be geographically dispersed and need not necessarily be concentrated. Knowledge of geographical location of each parking slot is assumed to be available to anyone in the system. A *trustworthy central controller* (CC) is responsible for coordinating the parking slot assignment mechanism, namely it is the central authority. The coordinations are carried out through secured channels.

The parking slot assignment mechanism is assumed to operate in slotted time, with the slots normalized to integer values  $t \in \{1, 2, 3, \dots\}$ . At the beginning of every time slot  $t$ , the set  $\mathcal{M}_t \subseteq \mathcal{M}$  of *free* parking slots is known.<sup>1</sup> In addition, at the

<sup>1</sup>Such information is retrieved by installing sensors at every parking slots.

(a)					
Car index $i$	Free Parking Slot, $j$				
	$j = 1$	2	3	4	5
1	0	1	0	0	0
2	0	0	1	0	0
3	0	1	0	0	0

(b)					
Car index $i$	Free Parking Slot, $j$				
	$j = 1$	2	3	4	5
1	0	1	0	0	0
2	0	0	1	0	0
3	1	0	0	0	0

TABLE 1: Assignment:  $\mathcal{N}_t = \{1, 2, 3\}$ ,  $\mathcal{M}_t = \{1, 2, 3, 4, 5\}$ : (a) An infeasible assignment; (b) A feasible assignment.

beginning of every time slot  $t$ , a set  $\mathcal{N}_t = \{1, \dots, N_t\}$  of cars is scheduled for parking slot assignment, where  $N_t \leq |\mathcal{M}_t|$  is the total number of cars. We denote by  $\text{des}(i)$  the destination of car  $i \in \mathcal{N}_t$  and by  $d_{ij}$  the distance from  $\text{des}(i)$  to free parking slot  $j \in \mathcal{M}_t$ . We assume that each car  $i$  can compute  $\{d_{ij}\}_{j \in \mathcal{M}_t}$  simply by knowing the geographical location of  $\text{des}(i)$ . Such computations can easily be performed by using the state-of-the-art global positioning system (GPS).

To formally express the problem, let us first introduce *binary decision variables*  $(x_{ij})_{i \in \mathcal{N}_t, j \in \mathcal{M}_t}$ , which indicate the  $i$  to  $j$  assignments as follows:

$$x_{ij} = \begin{cases} 1 & \text{car } i \text{ is assigned to parking slot } j \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

A *feasible assignment* should be such that one car is assigned to only one free parking slot and no more than one car is assigned to a free parking slot. For example, Table (a) shows an infeasible assignment and Table (b) shows a feasible assignment. Now we can formally express the distance from car  $i$ 's assigned parking slot to its destination  $\text{des}(i)$  as  $\sum_{j \in \mathcal{M}_t} d_{ij} x_{ij}$ . We refer to  $\sum_{j \in \mathcal{M}_t} d_{ij} x_{ij}$  as *parking distance* of car  $i$ .

In order to ensure min-max fairness among the cars, we *minimize the maximum parking distance*. Min-max fairness is appealing in many application domains in the sense that it ensures equalization of the costs incurred by the users, see [22]. Specifically, the problem can be formally expressed as

$$\text{minimize} \quad \max_{i \in \mathcal{N}_t} \sum_{j \in \mathcal{M}_t} d_{ij} x_{ij} \quad (2a)$$

$$\text{subject to} \quad \sum_{i \in \mathcal{N}_t} x_{ij} \leq 1, \quad j \in \mathcal{M}_t \quad (2b)$$

$$\sum_{j \in \mathcal{M}_t} x_{ij} = 1, \quad i \in \mathcal{N}_t \quad (2c)$$

$$x_{ij} \in \{0, 1\}, \quad i \in \mathcal{N}_t, j \in \mathcal{M}_t, \quad (2d)$$

where the variable is  $(x_{ij})_{i \in \mathcal{N}_t, j \in \mathcal{M}_t}$ . Constraint (2b) ensures that no more than one car is assigned to a free parking slot. Constraint (2c) imposes that each car is assigned to only one free slot. Finally, constraint (2d) ensures that the values of  $x_{ij}$  are either 0 or 1.

Note that the problem is nonconvex and even combinatorial. Hence we have to rely on global optimal methods [26] such as exhaustive search and branch and bound methods to *solve* it. The main disadvantage of global methods is the prohibitive computational complexity, even in the case of small problems. Such methods are not scalable, and therefore can be impractical. In the sequel, we provide a method based on duality. Even though the optimality cannot be guaranteed, the proposed method is efficient, fast, and allows distributed implementation with a little coordination from the CC.

### III. SOLUTION APPROACH VIA DUAL PROBLEM

In this section, we first equivalently formulate problem (2) in its epigraph form [23]. Then we apply duality theory to obtain the related dual problem, and show that the problem is split into subproblems and a master problem which can be solved efficiently.

The equivalent problem is given by <sup>2</sup>

$$\text{minimize} \quad s \quad (3a)$$

$$\text{subject to} \quad \sum_{j \in \mathcal{M}} d_{ij} x_{ij} \leq s, \quad i \in \mathcal{N} \quad (3b)$$

$$\sum_{i \in \mathcal{N}} x_{ij} \leq 1, \quad j \in \mathcal{M} \quad (3c)$$

$$\sum_{j \in \mathcal{M}} x_{ij} = 1, \quad i \in \mathcal{N} \quad (3d)$$

$$x_{ij} \in \{0, 1\}, \quad i \in \mathcal{N}, j \in \mathcal{M}, \quad (3e)$$

where the variables are  $s$  and  $\mathbf{x} = (x_{ij})_{i \in \mathcal{N}, j \in \mathcal{M}}$ . Note that like problem (2), (3) is still nonconvex. Now we seek to decouple the problem among the cars, in order to maintain scalability properties of the car parking mechanism. In this context, we can clearly see that constraints (3d), (3e) are already decoupled, yet constraints (3b), (3c) are coupled among the cars, which is an obstacle to distributed solution methods.

Let us now form the partial Lagrangian by dualizing the coupling constraints (3b) and (3c). To do this, we introduce multipliers  $\boldsymbol{\lambda} = (\lambda_i)_{i \in \mathcal{N}}$  for the inequality constraints (3b) and multipliers  $\boldsymbol{\mu} = (\mu_j)_{j \in \mathcal{M}}$  for the inequality constraints (3c). The Lagrangian associated with problem (3) is

$$\begin{aligned} L(s, \mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = & s + \sum_{i \in \mathcal{N}} \lambda_i \left( \sum_{j \in \mathcal{M}} d_{ij} x_{ij} - s \right) \\ & + \sum_{j \in \mathcal{M}} \mu_j \left( \sum_{i \in \mathcal{N}} x_{ij} - 1 \right) \\ = & s \left( 1 - \sum_{i \in \mathcal{N}} \lambda_i \right) + \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{M}} (\lambda_i d_{ij} + \mu_j) x_{ij} \\ & - \sum_{j \in \mathcal{M}} \mu_j. \end{aligned} \quad (4)$$

The dual function  $g(\boldsymbol{\lambda}, \boldsymbol{\mu})$  is given by

$$g(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \inf_{s \in \mathbb{R}, \sum_{j \in \mathcal{M}} x_{ij} = 1, i \in \mathcal{N}, x_{ij} \in \{0, 1\}, i \in \mathcal{N}, j \in \mathcal{M}} L(s, \mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \quad (5a)$$

$$= \begin{cases} \inf_{\sum_{j \in \mathcal{M}} x_{ij} = 1, i \in \mathcal{N}, \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{M}} (\lambda_i d_{ij} + \mu_j) x_{ij} - \sum_{j \in \mathcal{M}} \mu_j} & \text{if } \sum_{i \in \mathcal{N}} \lambda_i = 1 \\ -\infty & \text{otherwise} \end{cases} \quad (5b)$$

<sup>2</sup>Without loss of generality, we drop the subindex  $t$  for notational simplicity.

$$= \begin{cases} \sum_{i \in \mathcal{N}} \left( \inf_{\substack{\sum_{j \in \mathcal{M}} x_{ij} = 1, \\ x_{ij} \in \{0,1\}, j \in \mathcal{M}}} \sum_{j \in \mathcal{M}} (\lambda_i d_{ij} + \mu_j) x_{ij} \right) - \sum_{j \in \mathcal{M}} \mu_j & \text{if } \sum_{i \in \mathcal{N}} \lambda_i = 1 \\ -\infty & \text{otherwise} \end{cases} \quad (5c)$$

$$= \begin{cases} \sum_{i \in \mathcal{N}} g_i(\boldsymbol{\lambda}, \boldsymbol{\mu}) - \sum_{j \in \mathcal{M}} \mu_j & \sum_{i \in \mathcal{N}} \lambda_i = 1 \\ -\infty & \text{otherwise} \end{cases}, \quad (5d)$$

where the equality (5b) follows from that the linear function  $s(1 - \sum_{i \in \mathcal{N}} \lambda_i)$  is bounded below only when it is identically zero, the equality (5c) follows from that constraints (3d), (3e) are separable, and  $g_i(\boldsymbol{\lambda}, \boldsymbol{\mu})$  is the optimal value of the problem

$$\begin{aligned} & \text{minimize} && \sum_{j \in \mathcal{M}} (\lambda_i d_{ij} + \mu_j) x_{ij} \\ & \text{subject to} && \sum_{j \in \mathcal{M}} x_{ij} = 1 \\ & && x_{ij} \in \{0, 1\}, j \in \mathcal{M}, \end{aligned} \quad (6)$$

with the variable  $(x_{ij})_{j \in \mathcal{M}}$ . Note that problem (6) is a combinatorial problem. Nevertheless, it has a closed-form solution

$$x_{ij}^* = \begin{cases} 1 & j = \arg \min_{l \in \mathcal{M}} (\lambda_i d_{il} + \mu_l) \\ 0 & \text{otherwise} \end{cases}. \quad (7)$$

The dual master problem is

$$\text{maximize} \quad g(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \sum_{i \in \mathcal{N}} g_i(\boldsymbol{\lambda}, \boldsymbol{\mu}) \quad (8a)$$

$$\text{subject to} \quad \sum_{i \in \mathcal{N}} \lambda_i = 1 \quad (8b)$$

$$\lambda_i \geq 0, i \in \mathcal{N} \quad (8c)$$

$$\mu_j \geq 0, j \in \mathcal{M}, \quad (8d)$$

where the variables are  $\boldsymbol{\lambda}$  and  $\boldsymbol{\mu}$ . In the sequel, we describe an approach to solve the dual problem (8), based on the projected subgradient method [25].

#### Solving the dual

Note that  $g(\boldsymbol{\lambda}, \boldsymbol{\mu})$  is a concave function, therefore, we need to find the subgradient  $\mathbf{s} \in \mathbb{R}^{N+M}$  of  $-g$  at a feasible  $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ . For clarity we separate  $\mathbf{s}$  into two vectors as follows:

$$\mathbf{s} = (\mathbf{u}, \mathbf{v}), \quad (9)$$

where  $\mathbf{u} = (u_i)_{i \in \mathcal{N}}$  is the part of  $\mathbf{s}$  that corresponds to  $\boldsymbol{\lambda}$  and  $\mathbf{v} = (v_j)_{j \in \mathcal{M}}$  the part that corresponds to  $\boldsymbol{\mu}$ . The negative of dual function  $-g(\boldsymbol{\lambda}, \boldsymbol{\mu})$  is given by

$$-g(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \sum_{j \in \mathcal{M}} \mu_j - \sum_{j \in \mathcal{M}} \mu_j \sum_{i \in \mathcal{N}} x_{ij}^* - \sum_{i \in \mathcal{N}} \lambda_i \sum_{j \in \mathcal{M}} d_{ij} x_{ij}^*,$$

and particular choices for  $u_i, i \in \mathcal{N}$  and  $v_j, j \in \mathcal{M}$  are given by

$$u_i = - \sum_{j \in \mathcal{M}} d_{ij} x_{ij}^* \quad \text{and} \quad v_j = 1 - \sum_{i \in \mathcal{N}} x_{ij}^*. \quad (10)$$

Thus the projected subgradient method is given by

$$(\boldsymbol{\lambda}^{(k+1)}, \boldsymbol{\mu}^{(k+1)}) = P((\boldsymbol{\lambda}^{(k)}, \boldsymbol{\mu}^{(k)}) - \alpha_k (\mathbf{u}^{(k)}, \mathbf{v}^{(k)})), \quad (11)$$

where  $k$  is the current iteration index of the subgradient method,  $P(\mathbf{z})$  is the Euclidean projection of  $\mathbf{z} \in \mathbb{R}^{N+M}$  onto the *feasible set* of the dual problem (8), and  $\alpha_k > 0$  is the  $k$ th step size, chosen to guarantee the asymptotic convergence

of the subgradient method, e.g.,  $\alpha_k = \alpha/k$ , where  $\alpha$  is a positive scalar. Since the feasible set of dual problem is separable in  $\lambda$  and  $\mu$ , the projection  $P(\cdot)$  can be performed independently. Therefore, the iteration (11) is equivalently performed as follows:

$$\boldsymbol{\lambda}^{(k+1)} = P_s(\boldsymbol{\lambda}^{(k)} - \alpha_k \mathbf{u}^{(k)}) \quad (12)$$

$$\boldsymbol{\mu}^{(k+1)} = [\boldsymbol{\mu}^{(k)} - \alpha_k \mathbf{v}^{(k)}]^+, \quad (13)$$

where  $P_s(\cdot)$  is the Euclidean projection onto the probability simplex [27],

$$\Pi = \left\{ \boldsymbol{\lambda} \mid \sum_{i=1}^N \lambda_i = 1, \lambda_i \geq 0 \right\} \quad (14)$$

and  $[\cdot]^+$  is the Euclidean projection onto the nonnegative orthant. Note that the Euclidean projection onto the probability simplex can be posed as a convex optimization problem that can be solved efficiently, see Appendix A.

## IV. ALGORITHM DEVELOPMENT

In this section, we first present our distributed algorithm to address problem (2) via the dual problem (8). The resulting algorithm is indeed the distributed car parking mechanism that can be coordinated by the CC or the central controller. Next, we discuss the convergence properties of the algorithm.

### A. Distributed algorithm implementation

Roughly speaking, the algorithm capitalizes on the ability of the CC to construct the subgradient  $(\mathbf{u}, \mathbf{v})$  in a distributed fashion via the coordination of scheduled cars. Note that, the involvement of a CC (e.g., an authority who handles the parking slots) is essential for realizing the overall algorithm in practice. This involvement is mainly for coordinating certain parameter among the scheduled cars, and for constricting a feasible assignment in case the assignment from dual problem is infeasible. The algorithm is formally documented below, see also Fig. 2 for a concise block diagram.

---

*Algorithm:* DISTRIBUTED CAR-PARKING (DCP)

- 1) Given the distances  $(d_{ij})_{j \in \mathcal{M}}$  for each car  $i \in \mathcal{N}$ . The central controller (CC) sets  $k = 1$ , sets current objective value  $p^{\text{cur}}(0) = \infty$ , sets number of conflicting users  $N^{\text{conflict}} = N$ , and broadcasts the initial (feasible)  $\lambda_i^{(k)}$  and  $\boldsymbol{\mu}^{(k)} = (\mu_j^{(k)})_{j \in \mathcal{M}}$  to each car  $i \in \mathcal{N}$ .
- 2) Every car  $i$  sets  $\lambda_i = \lambda_i^{(k)}$  and  $\boldsymbol{\mu} = \boldsymbol{\mu}^{(k)}$  and locally computes  $\mathbf{x}_i^{(k)} = (x_{ij}^*)_{j \in \mathcal{M}}$  from (7). Let  $j_i^k$  denote the index of the nonzero component of  $\mathbf{x}_i^{(k)}$ .
- 3) Local subgradients: Each car  $i$ 
  - a. sets *scalar*  $u_i^{(k)} = - \sum_{j \in \mathcal{M}} d_{ij} x_{ij}^* = -d_{ij_i^k}$ , [compare with (10)].
  - b. transmits  $(u_i^{(k)}, j_i^k)$  to CC.
- 4) Current assignment and Subgradient iteration at CC
  - a. find the set  $\mathcal{J}_j^{(k)}$  of users assigned to slot  $j$ , i.e.,  $\mathcal{J}_j^{(k)} = \{i \mid j_i^k = j\}$ . Set  $N_k^{\text{conflict}} = \sum_{j \mid \text{card}(\mathcal{J}_j^{(k)}) \geq 2} \text{card}(\mathcal{J}_j^{(k)})$ .

- b. if no conflicting assignments (i.e.,  $N_k^{\text{conflict}} = 0$ ), set  $N^{\text{conflict}} = N_k^{\text{conflict}}$  and go to step 4-c. Otherwise, go to step 4-d.
  - c. if  $p^{\text{cur}}(k-1) > \max_{i \in \mathcal{N}} d_{ij_i^k}$ , set  $p^{\text{cur}}(k) = \max_{i \in \mathcal{N}} d_{ij_i^k}$  and set  $\mathbf{X}^{\text{cur}}(k) = (\mathbf{e}_{j_i^k}^T)_{i \in \mathcal{N}} \in \mathbb{R}^{N \times M}$ . Go to step 4-e.
  - d. if  $N_k^{\text{conflict}} < N^{\text{conflict}}$ , set  $N^{\text{conflict}} = N_k^{\text{conflict}}$ ,  $p^{\text{cur}}(k) = \infty$ ,  $\mathbf{X}^{\text{cur}}(k) = (\mathbf{e}_{j_i^k}^T)_{i \in \mathcal{N}} \in \mathbb{R}^{N \times M}$ , and  $\mathcal{J}_j^{\text{cur}} = \mathcal{J}_j^{(k)}$ ,  $j \in \mathcal{M}$ . Go to step 4-e.
  - e. form  $\mathbf{u}^{(k)} = (u_i^{(k)})_{i \in \mathcal{N}}$  and perform (12) to find  $\lambda^{(k+1)}$ .
  - f. set  $v_j = 1 - \text{card}(\mathcal{J}_j)$ ,  $\mathbf{v}^{(k)} = (v_j)_{j \in \mathcal{M}}$ , [compare with (10)].
  - g. perform (13) to find  $\mu^{(k+1)}$ .
- 5) Stopping criterion: If the stopping criterion is satisfied,
- a. go to step 6.
- Otherwise,
- b. CC broadcasts the new  $\lambda_i^{(k+1)}$  and  $\mu^{(k+1)}$  to each car  $i \in \mathcal{N}$ .
  - c. increment  $k$ , i.e., set  $k = k + 1$ .
  - d. go to step 2.
- 6) Output: If  $N^{\text{conflict}} = 0$  (i.e., a feasible assignment is achieved), CC returns  $\mathbf{X}^{\text{final}} = \mathbf{X}^{\text{cur}}(k)$  and terminates the algorithm. Otherwise, CC sets  $\mathbf{X}^{\text{infeasible}} = \mathbf{X}^{\text{cur}}(k)$ , performs a simple *subroutine* to construct a feasible assignment  $\mathbf{X}^{\text{final}}$  from  $\mathbf{X}^{\text{infeasible}}$ , returns  $\mathbf{X}^{\text{final}}$ , and terminates the algorithm.

### B. Algorithm description

In step 1, the algorithm starts by choosing initial feasible values for  $\lambda_i^{(k)}$ ,  $i \in \mathcal{N}$  and  $\mu_j^{(k)}$ ,  $j \in \mathcal{M}$ . Step 2 corresponds to the local computations of  $\mathbf{x}_i^{(k)}$  at each car  $i$ . These computations involve simple comparisons [see (7)] and can be performed in a parallelized manner by the scheduled cars. Step 3 involves coordination of scheduled cars and CC. First, each car  $i$  constructs *scalar* parameter  $u_i^{(k)}$ . Then it transmits  $u_i^{(k)}$  together with the potential car slot index  $j_i^k$  to CC.

In step 4, CC keeps records of the *best* assignment so far. The assignment is best, in the following sense. First suppose algorithm yields at least one feasible assignment, i.e.,  $N^{\text{conflict}} = 0$ . Then the best assignment is the one that corresponds to the smallest objective value among all feasible assignments, see step 4-c. On the other hand, suppose algorithm does not yield any feasible assignment, i.e.,  $N^{\text{conflict}} > 0$ . Then the best assignment is the one corresponds to the smallest  $N_k^{\text{conflict}}$  among all infeasible assignments, see step 4-d. Note that  $N^{\text{conflict}}$  is equal to the total conflicting users, and thus quantifies the degrees of infeasibility, see step 4-a. Moreover, by using the information received from scheduled cars, CC constructs the global subgradient components  $\mathbf{u}^{(k)} \in \mathbb{R}^N$  and  $\mathbf{v}^{(k)} \in \mathbb{R}^M$ , which in turn are used to perform the subgradient iterations (12)-(13), see steps 4-e,4-f,4-g.

The new parameters  $\lambda^{(k+1)}$  and  $\mu^{(k+1)}$  are broadcasted to every car and the algorithm is performed in an iterative manner until a stopping criterion is satisfied, see step 5. Natural

stopping criterion in practice includes running subgradient method for a fixed number of iterations.

Recall that, solution method for the primal problem (3) by considering its dual problem (8) does not always guarantee the primal feasibility, because the original problem (3) is *nonconvex* [27]. Therefore, if a feasible assignment is not achieved, a subroutine call is required to construct one after the stopping criterion is satisfied. Step 6 is essentially to address this infeasibility problem. In particular, once the stopping criterion is satisfied (step 5), CC checks whether the current assignment  $\mathbf{X}^{\text{cur}}(k)$  obtained is feasible. If it is feasible, algorithm terminates by returning  $\mathbf{X}^{\text{final}} = \mathbf{X}^{\text{cur}}(k)$ , where CC informs each car  $i$ , its parking slot. Otherwise, CC performs a simple subroutine to construct a feasible assignment  $\mathbf{X}^{\text{final}}$  by using the current infeasible assignment  $\mathbf{X}^{\text{infeasible}}$ , before the algorithm terminates. In the sequel, we outline a subroutine that can be implemented at CC for constructing a feasible assignment.

### C. Constructing a feasible assignment from $\mathbf{X}^{\text{infeasible}}$

The key idea of the subroutine is summarized as follows: 1) select the set of cars that are assigned to the *same* parking slot, 2) find the set of free parking slots, and 3) assign the conflicting cars found in the first stage to the free parking slots found in the second stage in an iterative manner. In the following, we describe this idea in the detail.

We start by introducing some useful notations for clarity. We denote by  $\mathcal{M}^{\text{over-assigned}}$  the set of parking slots, where two or more than two cars are assigned, i.e.,  $\mathcal{M}^{\text{over-assigned}} = \{j \mid \text{card}(\mathcal{J}_j^{\text{cur}}) \geq 2\}$ . Moreover, we denote by  $\mathcal{M}^{\text{free}}$  the set of free parking slots, i.e.,  $\mathcal{M}^{\text{free}} = \{j \in \mathcal{M} \mid \mathcal{J}_j^{\text{cur}} = \emptyset\}$ . For example, suppose Table (a) corresponds to the current assignment  $\mathbf{X}^{\text{infeasible}}$ , which is infeasible. Then we have  $\mathcal{M}^{\text{over-assigned}} = \{2\}$  and  $\mathcal{M}^{\text{free}} = \{1, 4, 5\}$ . To formally express the subroutine, it is further useful to introduce some minor notations, where we relabel the indices of cars and parking slots. Let  $\sigma = (\sigma_l)_{l=1, \dots, \text{card}(\mathcal{M}^{\text{over-assigned}})}$  denote the parking slot indices  $j \in \mathcal{M}^{\text{over-assigned}}$  arranged in an increasing order. Moreover, we denote by  $n_j$  the total cars assigned to  $j$ th parking slot [i.e.,  $\text{card}(\mathcal{J}_j^{\text{cur}})$ ], where  $j \in \mathcal{M}^{\text{over-assigned}}$ . Now, the subroutine can be formally expressed as follows:

**Algorithm:** CONSTRUCT A FEASIBLE ASSIGNMENT FROM  $\mathbf{X}^{\text{infeasible}}$

- 1) Given the infeasible assignment  $\mathbf{X}^{\text{infeasible}}$ ;  $\mathcal{M}^{\text{over-assigned}}$ ,  $\mathcal{M}^{\text{free}}$ ,  $\sigma$ , and  $n_j \forall j \in \mathcal{M}^{\text{over-assigned}}$ . Set  $\mathbf{X}^{\text{final}} = \mathbf{X}^{\text{infeasible}}$ ,  $k = 1$ , and  $l = 1$ .
- 2) CC sets  $\pi = (\pi_n)_{n=1, \dots, n_{\sigma_l}}$  to be the car indices  $i \in \mathcal{J}_{\sigma_l}^{\text{cur}}$  arranged in an increasing order.
- 3) For  $n = 2 : n_{\sigma_l}$ 
  - a. CC sends  $\mathcal{M}^{\text{free}}$  to car  $\pi_n$ .
  - b. car  $\pi_n$  chooses slot  $j$ , where  $j = \arg \min_j d_{\pi_n j}$  and sends  $j$  to CC.
  - c. CC updates  $\mathcal{M}^{\text{free}}$  as  $\mathcal{M}^{\text{free}} = \mathcal{M}^{\text{free}} \setminus \{j\}$  and sets  $[\mathbf{X}^{\text{final}}]_{\pi_n j} = 1$ .
- 4) If  $l = \text{card}(\mathcal{M}^{\text{over-assigned}})$ , return  $\mathbf{X}^{\text{final}}$  and STOP. Otherwise, set  $l = l + 1$  and go to step 2.

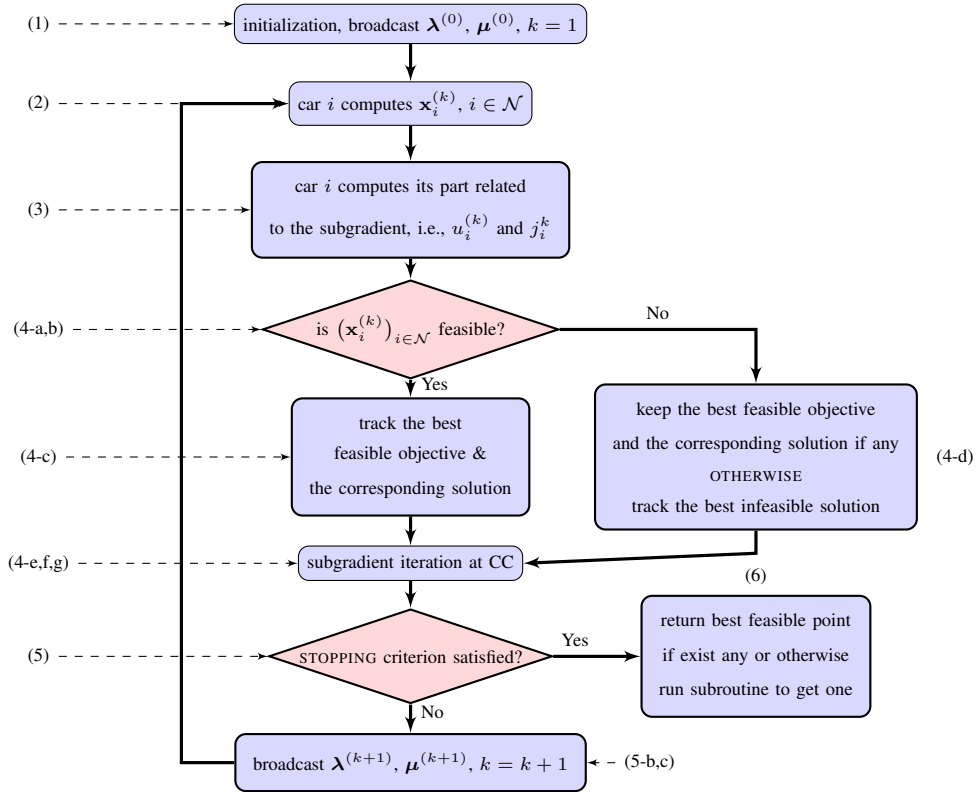


Fig. 2: Block diagram of the DCP algorithm.

Step 1 is the initialization of the subroutine. Step 2 takes every over-assigned parking slots in the order defined by  $\sigma$  and the corresponding conflicting car indices are ordered as  $\pi$ . In step 3, all of these cars, but  $\pi_1$  are assigned to free parking slots in an iterative manner. In particular, the conflicting cars (except  $\pi_1$ ) in the given over-assigned parking slot are given the free parking slot indices (see step 3-a) and every car chooses its parking slots in a greedy manner (see step 3-b). Note that the assignment of car  $\pi_1$  is not changed and it can remain in the slot already specified by  $\mathbf{X}^{\text{final}}$ , because the other cars are reassigned in step 3-a and step 3-b. Moreover, CC updates the assignment  $\mathbf{X}^{\text{final}}$  accordingly, see step 3-c. Stopping criterion in step 4 checks whether all the cars in the over-assigned parking slots have been reassigned. If so, subroutine terminates by returning the feasible assignment  $\mathbf{X}^{\text{final}}$ . Otherwise, the subroutine continues by moving to the next over-assign parking slot. If the subroutine above is applied to  $\mathbf{X}^{\text{infeasible}}$  given in Table (a), then a possible feasible assignment  $\mathbf{X}^{\text{final}}$  is shown in Table (b).

#### D. Convergence

In this section, we present the convergence properties of the proposed DCP algorithm for car parking. In particular, we show that for a sufficiently large number of subgradient iterations, the DCP algorithm converges to the dual optimal value of problem (8). The convergence is established by the following proposition:

**Proposition 1:** Let  $g_{\text{best}}^{(k)} = \max\{g(\lambda^{(1)}, \mu^{(1)}), \dots, g(\lambda^{(k)}, \mu^{(k)})\}$  denote the dual objective value found after  $k$  subgradient iterations and

$(\lambda^*, \mu^*)$  denote the optimal solution of dual problem (8). Suppose  $\|(\lambda^{(1)}, \mu^{(1)}) - (\lambda^*, \mu^*)\|$  is bounded from above. Then,  $\forall \varepsilon > 0, \exists n \geq 1$  such that  $\forall k \geq n \Rightarrow (d^* - g_{\text{best}}^{(k)}) < \varepsilon$ , where  $d^*$  is the optimal value of the dual problem (8).

*Proof:* The proof is based on the approach of [25], [27]. We have

$$\|(\lambda^{(k+1)}, \mu^{(k+1)}) - (\lambda^*, \mu^*)\|_2^2 = \|P((\lambda^{(k)} - \alpha_k \mathbf{u}^{(k)}) - \lambda^*, (\mu^{(k)} - \alpha_k \mathbf{v}^{(k)}) - \mu^*)\|_2^2 \quad (15)$$

$$\leq \|((\lambda^{(k)} - \alpha_k \mathbf{u}^{(k)}) - \lambda^*, (\mu^{(k)} - \alpha_k \mathbf{v}^{(k)}) - \mu^*)\|_2^2 \quad (16)$$

$$= \|(\lambda^{(k)}, \mu^{(k)}) - (\lambda^*, \mu^*)\|_2^2 - 2\alpha_k \mathbf{u}^{(k)T}(\lambda^{(k)} - \lambda^*) - 2\alpha_k \mathbf{v}^{(k)T}(\mu^{(k)} - \mu^*) + \alpha_k^2 \|\mathbf{u}^{(k)}\|_2^2 + \alpha_k^2 \|\mathbf{v}^{(k)}\|_2^2 \quad (17)$$

$$\leq \|(\lambda^{(k)}, \mu^{(k)}) - (\lambda^*, \mu^*)\|_2^2 - 2\alpha_k (g(\lambda^*, \mu^*) - g(\lambda^{(k)}, \mu^{(k)})) + \alpha_k^2 \|\mathbf{u}^{(k)}\|_2^2 + \alpha_k^2 \|\mathbf{v}^{(k)}\|_2^2, \quad (18)$$

where (15) follows from (11), (16) follows from that the Euclidean projection  $P(\mathbf{z})$  of any  $\mathbf{z} \in \mathbb{R}^{N+M}$  onto the *feasible set* of the dual problem (8) always decreases the distance of  $P(\mathbf{z})$  to every point in the *feasible set* and in particular to the optimal point  $(\lambda^*, \mu^*)$ , and (18) follows from the definition of subgradient. Recursively applying (18) and rearranging the terms we obtain

$$2 \sum_{l=1:k} \alpha_l (d^* - g(\lambda^{(l)}, \mu^{(l)})) \leq \|(\lambda^{(k+1)}, \mu^{(k+1)}) - (\lambda^*, \mu^*)\|_2^2 + \|(\lambda^{(1)}, \mu^{(1)}) - (\lambda^*, \mu^*)\|_2^2 + \sum_{l=1:k} \alpha_l^2 \|\mathbf{u}^{(l)}\|_2^2 + \sum_{l=1:k} \alpha_l^2 \|\mathbf{v}^{(l)}\|_2^2 \quad (19)$$

$$\leq \|(\lambda^{(1)}, \mu^{(1)}) - (\lambda^*, \mu^*)\|_2^2 + \sum_{l=1:k} \alpha_l^2 (\|\mathbf{u}^{(l)}\|_2^2 + \|\mathbf{v}^{(l)}\|_2^2) \quad (20)$$



$$\leq R^2 + G_1^2 \sum_{l=1:k} \alpha_l^2 + G_2^2 \sum_{l=1:k} \alpha_l^2 = R^2 + (G_1^2 + G_2^2) \sum_{l=1:k} \alpha_l^2, \quad (21)$$

where (20) follows from that  $\|(\lambda^{(k+1)}, \mu^{(k+1)}) - (\lambda^*, \mu^*)\|_2^2 \geq 0$ , (21) follows from that  $\|(\lambda^{(1)}, \mu^{(1)}) - (\lambda^*, \mu^*)\|$  is bounded from above, i.e.,  $\exists R < \infty$  such that  $\|(\lambda^{(1)}, \mu^{(1)}) - (\lambda^*, \mu^*)\| < R$  and the norm of the subgradient  $(\mathbf{u}, \mathbf{v})$  is bounded from above as

$$\|\mathbf{u}\|_2 \leq G_1 = \sqrt{\sum_{i \in \mathcal{N}} (\max_{j \in \mathcal{M}} d_{ij})^2} \quad (22)$$

$$\|\mathbf{v}\|_2 \leq G_2 = \sqrt{(N-1)^2 + (M-1)}. \quad (23)$$

The bound (22) is obtained by noting that there exist at most one nonzero element in  $(x_{ij})_{j \in \mathcal{M}}$ , see (7) and (10). Moreover, (23) follows when all cars are assigned to one parking slot, see (10). By using the trivial relation

$$d^* - g_{\text{best}}^{(k)} \leq d^* - g(\lambda^{(l)}, \mu^{(l)}), \quad l = 1, \dots, k, \quad (24)$$

and (21), we obtain an upper bound on  $d^* - g_{\text{best}}^{(k)}$  as

$$d^* - g_{\text{best}}^{(k)} \leq (R^2 + (G_1^2 + G_2^2) \sum_{l=1}^k \alpha_l^2) / (2 \sum_{l=1}^k \alpha_l) \quad (25)$$

Noting that step size  $\alpha_l = \alpha/l$ ,  $0 < \alpha < \infty$  is *square summable*, i.e.,  $\sum_{l=1}^{\infty} \alpha_l^2 = \alpha^2 \pi/6$ . Moreover,  $\sum_{l=1}^k \alpha_l$  is strictly monotonically increasing in  $k$  (it grows without bound as  $k \rightarrow \infty$ ). Therefore, for any  $\epsilon > 0$ , we can always find an integer  $n \geq 1$  such that  $\sum_{l=1}^k \alpha_l > \epsilon (R^2/2 + \alpha^2(G_1^2 + G_2^2)\pi/12)$  if  $k \geq n$ , which concludes the proof. ■

The bound derived in (25), together with (22)-(23) allows us to predict some key behaviors of the convergence of the proposed algorithm. For example, the larger the  $d_{ij}$  values, the larger the  $G_1$ , and therefore, the larger the number of iterations to achieve a given accuracy. Nevertheless, the influence of  $G_1$  can be made negligible by arbitrarily scaling down the objective function of problem (2). From (23), we note that the number of scheduled cars (i.e.,  $N$ ) and the number of free parking slots (i.e.,  $M$ ) directly influence the convergence. In the next section, we highlight some appealing privacy preserving properties of the DCP algorithm.

## V. PRIVACY PROPERTIES OF THE ALGORITHM

We see that the proposed car parking mechanism DCP is preserving privacy in the sense that any car  $n \neq i$  will not be able to find out the destination  $\text{des}(i)$  of  $i$ th car while using the DCP algorithm. We refer to an attempt of an arbitrary car  $n$  to discover the destination of any other car  $i$ , as a passive attack [28, § 5.1-5.3], where car  $n$  keeps records of possibly all the information that it exchanges with CC and by using those it tries to discover private data  $\text{des}(i)$ . In what follows, we first present sufficient information that an arbitrary car  $n$  can use to discover  $\text{des}(i)$ . Then we show how DCP algorithm cancels such a sufficient information and ensures privacy.

### A. Sufficient information to discover the destination

Let us first fix the adversary to be car  $n$  and assume that car  $n$  wants to discover  $\text{des}(1)$ , i.e., the destination of car 1. Now suppose car  $n$  knows the set  $\mathcal{C}_1 = \{(j_1^k, d_{1j_1^k})\}_{k=1,2,\dots,K}$

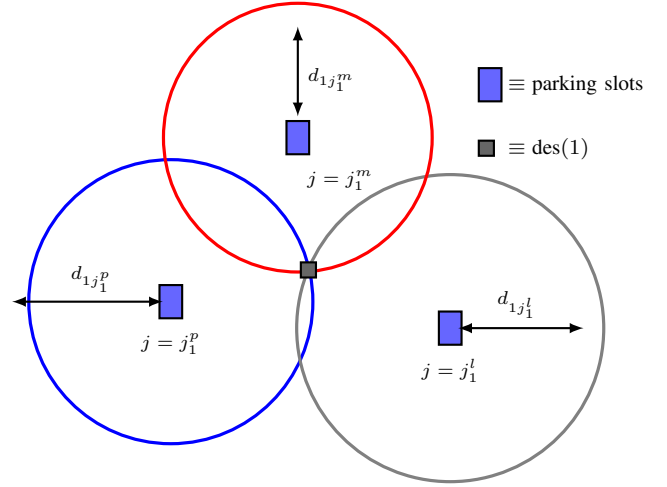


Fig. 3: Given  $(j_1^p, d_{1j_1^p})$ ,  $(j_1^l, d_{1j_1^l})$ ,  $(j_1^m, d_{1j_1^m})$  pairs known to the adversary (car  $n$ ),  $j_1^p \neq j_1^l$ ,  $j_1^p \neq j_1^m$ , and  $j_1^l \neq j_1^m$ , discovering the location of  $\text{des}(1)$ .

of data associated with car 1, where  $K$  is the total iterations of DCP algorithm. Provided there exists at least three distinct  $j_1^k$ s, car  $n$  can simply locate  $\text{des}(1)$  as illustrated in Fig. 3. Even if car  $n$  knows only the set  $\mathcal{D}_1 = \{d_{1j_1^k}\}_{k=1,2,\dots,K}$  of data associated with car 1, it turns out that car  $n$  can locate  $\text{des}(1)$  exhaustively. In particular, in every iteration  $k$ , car  $n$  draws  $M-1$  circles with radius  $d_{1j_1^k}$  centered at parking slots  $\mathcal{M} \setminus \{j_1^k\}$ . Let  $\mathcal{S}^k$  denote the aforementioned set of circles. Provided there are at least three distinct  $j_1^k$ s, which correspond to some iteration indexes  $l, m$ , and  $p$ , one can see that there exists at least one point at which a circle in  $\mathcal{S}^l$ , a circle in  $\mathcal{S}^m$ , and a circle in  $\mathcal{S}^p$  intersect. If this point is unique, then it corresponds to  $\text{des}(i)$ .<sup>3</sup> The discussion above indicates that if the adversary (car  $n$ ) knows  $\mathcal{C}_1$  or even  $\mathcal{D}_1$ , under mild conditions, it can locate  $\text{des}(i)$ . In the sequel, we show how DCP precludes such situations. In particular, we show how  $\{d_{1j_1^k}\}_{k=1,2,\dots,K}$  is kept hidden from the adversary car  $n$ .

### B. How to preserve privacy

Note that the only means by which car  $n$  gets access to some functions of  $\{d_{1j_1^k}\}_{k=1,2,\dots,K}$  is via  $\{\lambda_n^{(k)}\}_{k=1,2,\dots,K}$ , see step 5 of DCP algorithm. In other words, the involvement of car  $n$  during the DCP algorithm is restricted so that, in every iteration  $k$ , it has access to only some *interface* variables  $\lambda_n^{(k)}$  and  $\mu^{(k)}$ .<sup>4</sup> This restriction is indeed achieved by the decomposition structure of problem (2). Moreover, we consider the situation that CC uses the step size  $\alpha_k$  of DCP as

$$\alpha_k = \frac{\alpha}{k}, \quad (26)$$

where  $\alpha$  is arbitrarily chosen on  $[\alpha^{\min}, \alpha^{\max}]$ ,  $\alpha^{\min}$  and  $\alpha^{\max}$  are positive numbers known only to CC such that  $\alpha^{\min} < \alpha^{\max}$ . Note that the above choice of  $\alpha_k$  still preserves the

<sup>3</sup>When the parking slots are not arbitrarily located and there are symmetric properties, then there can be more than one intersection point, which in turn will create uncertainties in correctly locating  $\text{des}(i)$ .

<sup>4</sup>The knowledge of  $\mu^{(k)}$  is irrelevant here because it does not carry any information of  $d_{1j_1^k}$ .

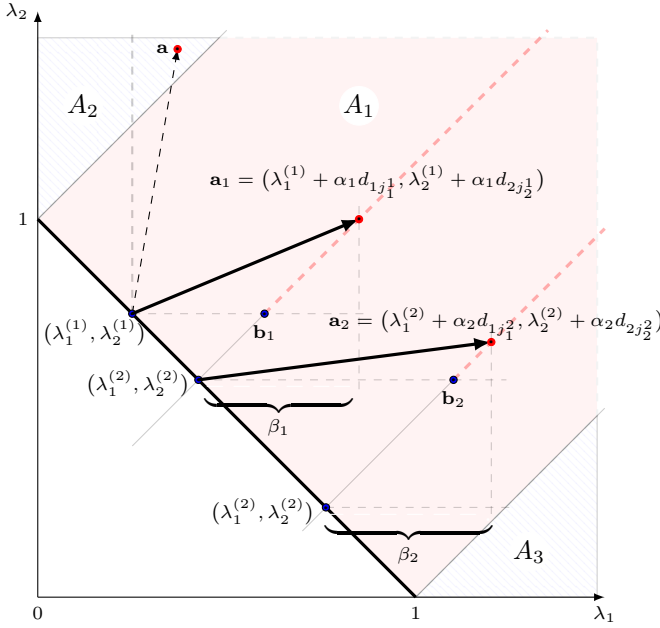


Fig. 4: The evolution of  $\lambda^{(k)} = (\lambda_1^{(k)}, \lambda_2^{(k)})$  in a case of 2-cars.

convergence properties established in *Proposition 1* (see Section IV-D) for the following reasons [compare with (25)]:

- 1)  $\sum_{k=1}^{\infty} \alpha_k \rightarrow \infty$ . This result is shown by noting that  $\sum_{k=1}^{\infty} \alpha^{\min}/k \rightarrow \infty$  and  $0 < \alpha^{\min}/k \leq \alpha_k$  for all  $k$ .
- 2)  $\sum_{k=1}^{\infty} \alpha_k^2$  converges to a point on  $[(\alpha^{\min})^2\pi/6, (\alpha^{\max})^2\pi/6]$ . This result is achieved by noting that  $\sum_{k=1}^{\infty} (\alpha^{\min})^2/k^2 \rightarrow (\alpha^{\min})^2\pi/6$ ,  $\sum_{k=1}^{\infty} (\alpha^{\max})^2/k^2 \rightarrow (\alpha^{\max})^2\pi/6$ , and  $(\alpha^{\min})^2/k^2 \leq \alpha_k^2 \leq (\alpha^{\max})^2/k^2$  for all  $k$ .

The arbitrary step size above (i.e., 26) essentially introduces more protection to the problem data  $\{d_{1j_1^k}\}_{k=1,2,\dots,K}$ .

Now we pose the following question: Does the proposed DCP allow car  $n$  alone to make records of  $\{d_{1j_1^k}\}_{k=1,2,\dots,K}$ , so that it can locate  $\text{des}(1)$  as discussed in Section V-A? It turns out that even though, car  $n$  can document the connections among the unknown parameters including  $\{d_{1j_1^k}\}_{k=1,2,\dots,K}$ , among others, it can only come up with an *under determined set of nonlinear equations*. Therefore  $\{d_{1j_1^k}\}_{k=1,2,\dots,K}$  cannot be computed as we will see next.

We consider only the case with  $n = 2$  and suppose car 2 is the adversary that wants to discover the destination of car 1, i.e.,  $\text{des}(1)$ . The discussion can be generalized to scenarios with  $n > 2$ , in a straightforward manner.

First, note that CC performs the projection of  $\lambda^{(k)} - \alpha_k \mathbf{u}^{(k)}$  onto the probability simplex to yield  $\lambda^{(k+1)}$  [see step 4-c of DCP algorithm]. In the considered 2-car case, the probability simplex is the line segment from  $(0, 1)$  to  $(1, 0)$ , see Fig. 4. Once car 2 is given  $\lambda_2^{(1)}$ , it can locate  $(\lambda_1^{(1)}, \lambda_2^{(1)})$ , because  $\lambda_1^{(1)} = 1 - \lambda_2^{(1)}$ . Yet car 2 cannot locate  $\mathbf{a}_1 = \lambda^{(1)} - \alpha_1 \mathbf{u}^{(1)}$ . After receiving  $\lambda_2^{(2)}$ , car 2 can locate  $(\lambda_1^{(2)}, \lambda_2^{(2)})$ . It can also locate  $\mathbf{a}_1$  up to the ray originating at  $\mathbf{b}_1$ , see Fig. 4. However, car 2 cannot exactly locate  $\mathbf{a}_1$ . The algorithm continues in a similar manner. For example, the evolution of  $\lambda^{(k)} = (\lambda_1^{(k)}, \lambda_2^{(k)})$  is illustrated in Fig. 4 for  $k = 1, 2$ , and 3. With this knowledge of  $\lambda^{(k)}$  evolution, car 2 can write a set

of equations in every iteration  $k$  as given in Table 2. Note that the set of equations for  $k > 1$  are nonlinear, because there are products of unknowns, e.g., (2.2), (2.3), (3.2), and (3.3).

When documenting the relations of unknowns in Table 2, we assume  $\{\mathbf{a}_k\}_{k=1,2,3,\dots}$  lies only in the shaded area  $A_1$ . In contrast, suppose  $\mathbf{a}_k$  lies either in the hatched area  $A_2$  or  $A_3$  for some iterations. One such point is depicted in Fig. 4, where  $\mathbf{a}_1 = \mathbf{a}$ , see the hatched area  $A_2$ . In this case,  $(\lambda_1^{(2)}, \lambda_2^{(2)})$  will be  $(0, 1)$ . Consequently, car 2 can locate  $\mathbf{a}_1$  only up to a cone instead of a ray, which in turn accounts for more uncertainties in determining  $\mathbf{a}_1$ . Such situations can only increase the difference between the number of unknowns ( $U_k$ ) and the number of equations ( $E_k$ ). For example, in this case, we will have  $U_2 - E_2 > 1$ , instead of  $U_2 - E_2 = 1$  as in Table 2.

Thus, we can conclude that, the total unknowns are always greater than the total equations. This results an under determined set of nonlinear equations, see Table 2. Therefore, car 2 cannot make records of unknowns  $\{d_{1j_1^k}\}_{k=1,2,\dots,K}$  and consequently it cannot discover  $\text{des}(1)$  as discussed in Section V-A.

## VI. NUMERICAL RESULTS

In this section we present the numerical evaluation of our proposed algorithm DCP. We compare the DCP algorithm to the following benchmarks:

- (a) Greedy parking policy: In this case, each car selects the closest parking slot to its destination.
- (b) Optimal parking policy: A solution of the optimization problem (3) is found by using the general solver, the IBM CPLEX optimizer [29].

In each time slot, the proposed algorithm is carried out for  $K$  subgradient iterations. In fact,  $K$  is used to define the stopping criterion at step 5 of DCP algorithm, see Fig. 2. In addition, the greedy policy and the optimal policy have also been performed at every time slot. We average the results over  $T$  time slots to demonstrate the average performances of the DCP algorithm. Specifically, at the beginning of every time slot, the total number of cars  $N$  and the total number of free parking slots  $M$  are considered to be fixed and the distances  $\{d_{ij}\}_{i \in \mathcal{N}, j \in \mathcal{M}}$  are considered uniformly distributed on  $[0, 1000]$ . The parking distances,  $\{d_{ij}\}_{i \in \mathcal{N}, j \in \mathcal{M}}$  are changed from slot to slot.

To simplify the presentation, we denote by  $p^{\text{cur}}(t, k)$  the best objective value achieved at time slot  $t$  after  $k$  subgradient iterations [compare to  $p^{\text{cur}}(k)$  in step 4-c,d of the DCP algorithm]. In particular,

$$p^{\text{cur}}(t, k) = \arg \min_{l=1,\dots,k} p(t, l), \quad (27)$$

where  $p(t, l)$  is the objective value at time slot  $t$  and at subgradient iteration  $l$ . Note that  $p^{\text{cur}}(t, k)$  is similar to  $p^{\text{cur}}(k)$  of DCP algorithm with an additional index  $t$  to indicate the time slot. Moreover, we denote by  $\mathbf{X}^{\text{cur}}(t, k)$  the best feasible or infeasible solution, which is identical to  $\mathbf{X}^{\text{cur}}(k)$  of DCP algorithm with an additional index  $t$  to indicate the time slot.

In all the considered simulations, we use  $T = 1000$ . Moreover,  $K$  is chosen to be 300 or 500. To simplify the



iteration ( $k$ )	relations of unknown parameters	unknowns	no. of unknowns ( $U_k$ )	no. of equations ( $E_k$ )
1	$\lambda_1^{(1)} + \lambda_2^{(1)} = 1 \rightarrow (1.1)$	$\lambda_1^{(1)}$	1	1
2	$(1.1)$ $\lambda_1^{(2)} + \lambda_2^{(2)} = 1 \rightarrow (2.1)$ $\lambda_1^{(2)} + \beta_1 = \lambda_1^{(1)} + \alpha_1 d_{1j_1^1} \rightarrow (2.2)$ $\lambda_2^{(2)} + \beta_1 = \lambda_2^{(1)} + \alpha_1 d_{2j_2^1} \rightarrow (2.3)$	$\lambda_1^{(1)}, \lambda_1^{(2)}$ $\beta_1$ $\alpha_1$ $d_{1j_1^1}$	5	4
3	$(1.1), (2.1), (2.2), (2.3)$ $\lambda_1^{(3)} + \lambda_2^{(3)} = 1 \rightarrow (3.1)$ $\lambda_1^{(3)} + \beta_2 = \lambda_1^{(2)} + \alpha_2 d_{1j_1^2} \rightarrow (3.2)$ $\lambda_2^{(3)} + \beta_2 = \lambda_2^{(2)} + \alpha_2 d_{2j_2^2} \rightarrow (3.3)$	$\lambda_1^{(1)}, \lambda_1^{(2)}, \lambda_1^{(3)}$ $\beta_1, \beta_2$ $\alpha_1, \alpha_2$ $d_{1j_1^1}, d_{1j_1^2}$	9	7
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

TABLE 2: Relations of unknown parameters as seen by the adversary car 2.

presentation, we refer to problem setups with  $N/M \leq 0.5$  as *lightly loaded* cases and refer to problem setups with  $N/M \geq 0.5$  as *heavily loaded* cases. Moreover, the problem setups with  $N/M \simeq 0.5$  are referred to as *moderately loaded* cases.

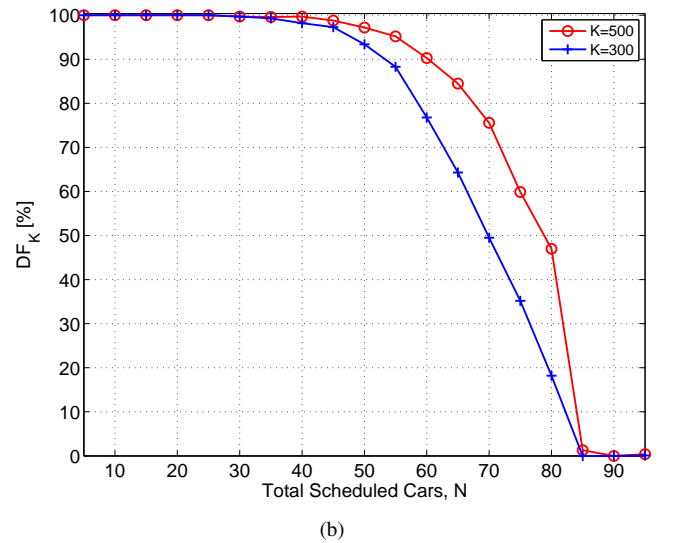
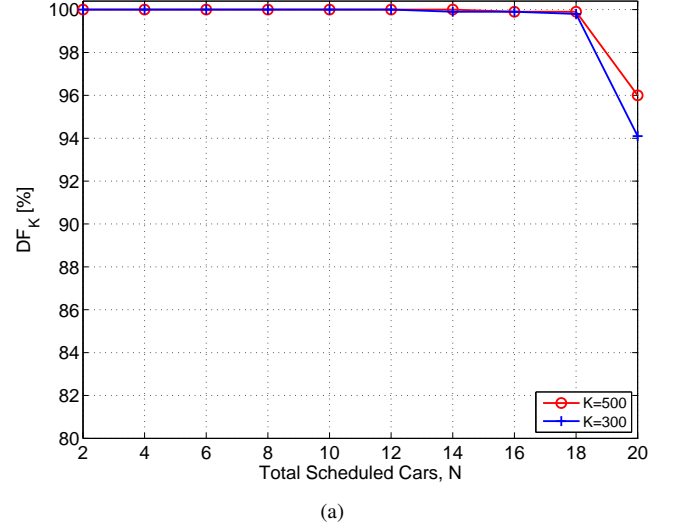
We first define a performance metric called *the degree of feasibility* of  $\mathbf{X}^{\text{cur}}(t, k)$ . Note that  $\mathbf{X}^{\text{cur}}(t, k)$  is, in fact, the assignment at the beginning of step 6 of DCP algorithm, which can be either feasible or infeasible. If  $\mathbf{X}^{\text{cur}}(t, k)$  is feasible, we have  $N^{\text{conflict}} = 0$  or equivalently,  $p^{\text{cur}}(t, K) < \infty$  [compare with step 6, 4-c, and 4-d]. On the other hand, if  $\mathbf{X}^{\text{cur}}(t, k)$  is infeasible, we have  $N^{\text{conflict}} > 0$  or equivalently,  $p^{\text{cur}}(t, K) = \infty$  [compare with step 6, 4-c, and 4-d]. This motivates to define the degree of feasibility (DF) of  $\mathbf{X}^{\text{cur}}(t, k)$  as

$$\text{DF}_K = \frac{\sum_{t=1}^T I(p^{\text{cur}}(t, K) < \infty)}{T} \times 100\%, \quad (28)$$

where  $I(E)$  is the indicator function of event  $E$ , i.e.,  $I(E) = 1$  if  $E$  is true or  $I(E) = 0$  otherwise.

Fig. 5 shows  $\text{DF}_K$  versus  $N$  for fixed  $M$ . In particular a smaller dimensional problem with  $M = 20$  [Fig. 5(a)] and a larger dimensional problem with  $M = 100$  [Fig. 5(b)] is considered. Results show that when  $N$  is significantly smaller than  $M$ , the degree of feasibility is almost 100%. Results further show that as  $N$  becomes closer to  $M$ , the degree of feasibility starts deteriorating. Moreover, it becomes significantly low in the case of larger dimensional problem [Fig. 5(b)] compared with the smaller dimensional problem [Fig. 5(a)]. For example, when  $N = M$ , algorithm yields  $\text{DF}_K$  values in the range 94–96% for the smaller dimensional problem setup. However, in the case of larger dimensional problem, when  $N = M$ , the degree of feasibility is almost zero. Not surprisingly, running DCP algorithm for larger number of subgradient iterations (e.g.,  $K = 500$ ) yields better feasibility results compared with smaller number of subgradient iterations (e.g.,  $K = 300$ ). However, the performance gap has been pronounced in the case of  $M = 100$  [Fig. 5(b)] compared to the case  $M = 20$ .

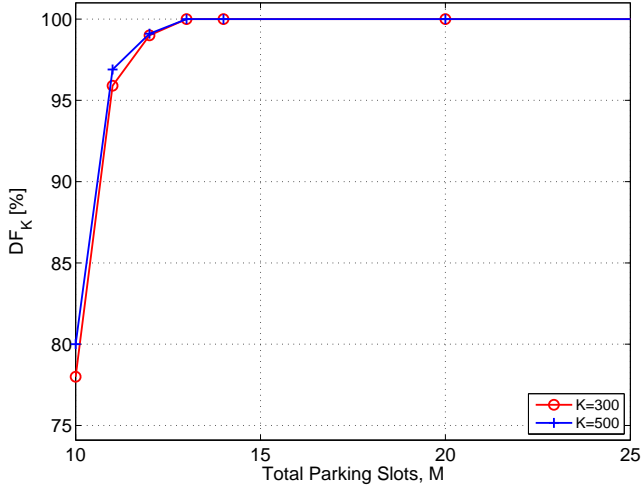
Fig. 6 shows  $\text{DF}_K$  versus  $M$  for fixed  $N$ . Again a smaller dimensional problem ( $N = 10$ ) and a larger dimensional problem with  $N = 50$  is considered, see Fig. 6(a)] and Fig. 6(b), respectively. Results resemble the observations of Fig. 5, where a desirable feasibility is achieved when  $M$  is significantly larger than  $N$  and the performances are pronounced for smaller dimensional problems.

Fig. 5: Degree of feasibility  $\text{DF}_K$  versus total cars  $N$ : (a)  $M = 20$ ; (b)  $M = 100$ .

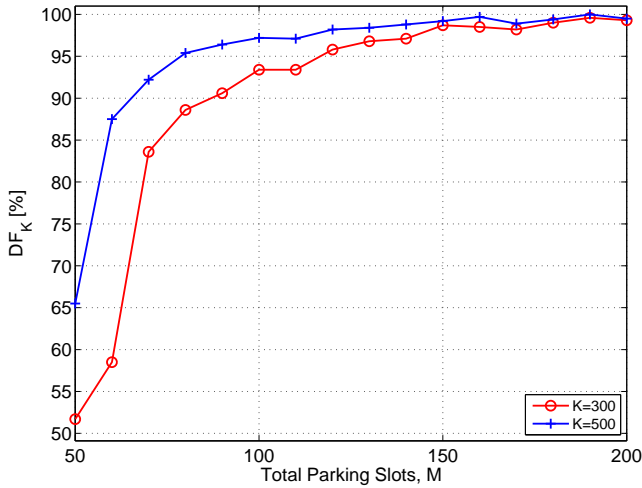
To see the average behavior of the DCP algorithm, now we consider the following performance metric, which is a measure of the average objective value at subgradient iteration  $k$ :

$$p^{\text{ave}}(k) = \frac{1}{T} \sum_{t=1}^T p^{\text{cur}}(t, k), \quad k = 1, \dots, K. \quad (29)$$

Fig. 7 shows  $p^{\text{ave}}(k)$  versus subgradient iterations  $k$  for cases  $M = 20$ ,  $N = 4$  [Fig. 7(a)] and  $M = 20$ ,



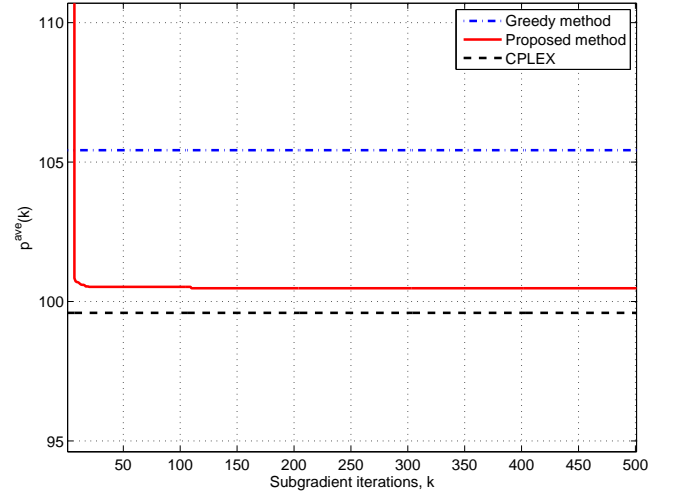
(a)



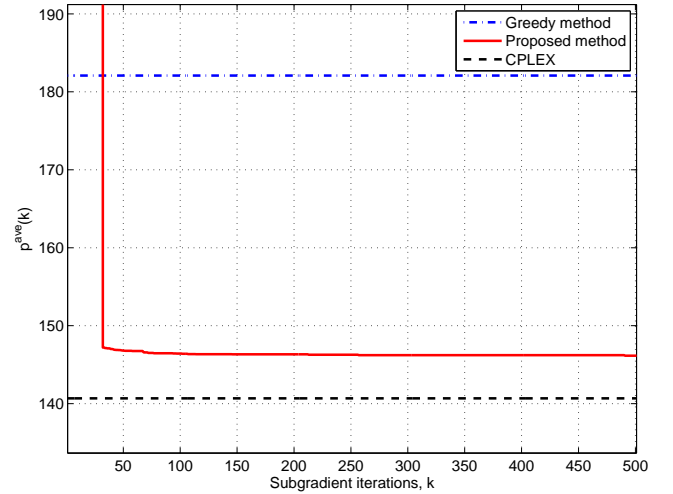
(b)

Fig. 6: Degree of feasibility  $DF_K$  vs total parking slots  $M$ : (a)  $N = 10$ ; (b)  $N = 50$ .

$N = 10$  [Fig. 7(b)]. Note that the vertical drops of the curves associated with our proposed method correspond to the subgradient iteration, before which a feasible assignment is found during any time slots  $t = \{1, \dots, T\}$ . Results show that in the lightly loaded case that corresponds to a smaller  $N$  (i.e.,  $N = 4$ ), a feasible assignment is found much earlier than the moderately loaded case which corresponds to  $N = 10$ . Specifically, when  $N = 4$ , the DCP algorithm yields a feasible assignment at most after  $k = 6$  subgradient iterations, whereas when  $N = 10$ , it yields a feasible assignment at most after  $k = 31$  subgradient iterations. This result is consistent with Fig. 5 and Fig. 6, because for fixed  $M$ , the higher the  $N$ , the higher the number of time slots among  $t = 1, \dots, T$  at which feasible solutions are achieved. For comparison, we also plot the average objective values obtained from the greedy method and the optimal CPLEX method. Not surprisingly, the optimal CPLEX method gives the best average objective, which is achieved at the expense of high computational complexity. However, our proposed method trades off an increase in average objective value for a low complexity in the algorithm, which gracefully scalable. Still the performance degradation



(a)

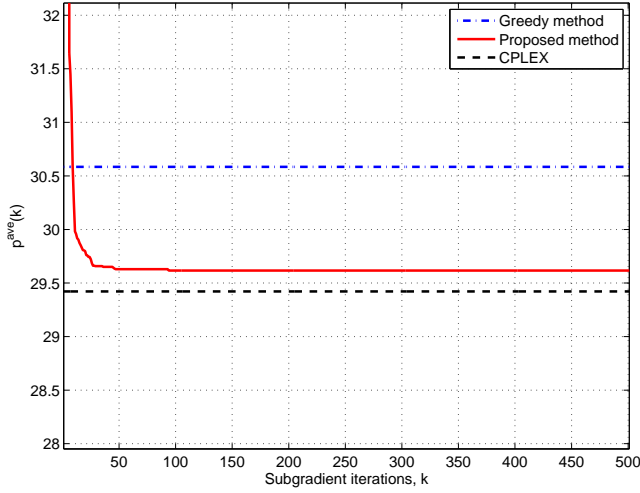


(b)

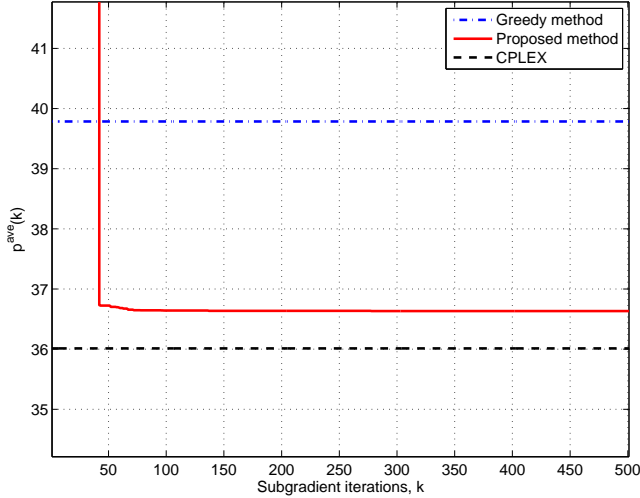
Fig. 7: Average objective  $p^{\text{ave}}(k)$  versus subgradient iterations  $k$ : (a)  $N = 4$  and  $M = 20$ ; (b)  $N = 10$  and  $M = 20$ .

of the proposed method is not critical. For example, the performance loss of DCP method compared with the optimal is 0.88% in the case of  $N = 4$  and 3.89% in the case of  $N = 10$ . It is interesting to note that the proposed DCP algorithm outperforms the greedy method. For example, the performance degradation of the greedy method compared with the optimal is 5.86% and 29.43% in the cases of  $N = 4$  and  $N = 10$ , respectively, which is significantly higher compared with our DCP method.

Fig. 8 shows  $p^{\text{ave}}(k)$  versus subgradient iterations  $k$  for a larger dimensional problem setup. In particular, we consider the cases  $M = 100$ ,  $N = 10$  [Fig. 8(a)] and  $M = 100$ ,  $N = 20$  [Fig. 8(b)]. The behavior of the plots are similar to those in Fig. 7. In the case of  $N = 10$ , the performance degradation of DCP method compared with the optimal is 0.66% and that of the greedy method is 3.95%. In the case of  $N = 20$ , the performance degradation of DCP method is 1.72% and that of the greedy method is 10.48%. Results thus show that even in larger networks our proposed DCP method can outperform the greedy approach substantially.



(a)

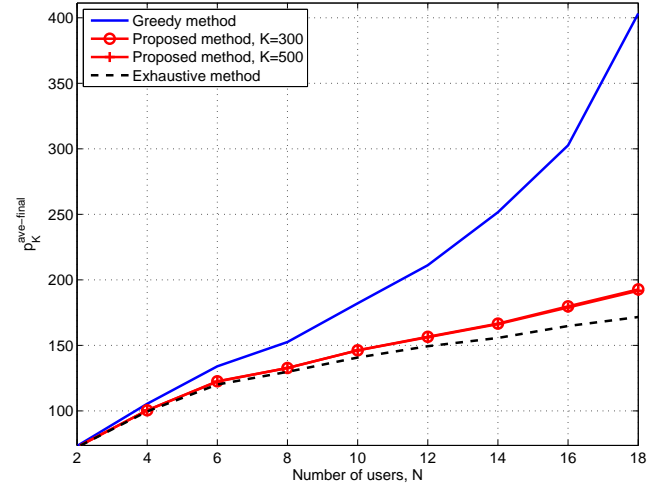


(b)

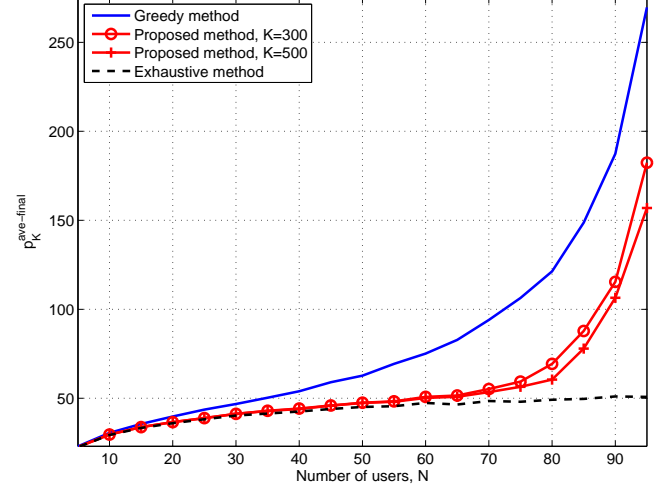
Fig. 8: Average objective  $p^{\text{ave}}(k)$  versus subgradient iterations  $k$ : (a)  $N = 10$  and  $M = 100$ ; (b)  $N = 20$  and  $M = 100$ .

As seen in Fig. 5 and Fig. 6, as the number of cars  $N$  becomes relatively closer to the total free parking slots  $M$  (i.e., heavily loaded scenarios), infeasibility of  $\mathbf{X}^{\text{cur}}(t, k)$  is usually inevitable. Therefore, as discussed in section IV-C, we have to rely on a subroutine to construct a feasible assignment, see step 6 in DCP algorithm. In the sequel, we show numerically the performance of proposed DCP algorithm in such heavily loaded cases.

Recall that, if  $p^{\text{cur}}(t, K) = \infty$ , then the corresponding assignment  $\mathbf{X}^{\text{cur}}(t, K)$  is infeasible. In such situations, our proposed algorithm invokes its subroutine (section IV-C) to construct a feasible assignment by using the best infeasible solution achieved so far, see step 6 of DCP. On the other hand, if  $p^{\text{cur}}(t, K) < \infty$ , then the corresponding assignment  $\mathbf{X}^{\text{cur}}(t, K)$  is already feasible. In either case, DCP algorithm returns a feasible point as given in step 6 of DCP algorithm. We denote by  $\mathbf{X}^{\text{final}}(t)$  this feasible assignment returned by our proposed DCP at time slot  $t$  and by  $p^{\text{final}}(t, K)$  the corresponding objective value. Finally, we denote by  $p_K^{\text{ave-final}}$  the average objective value achieved after the termination of



(a)



(b)

Fig. 9: Average objective value after the termination of DCP  $p_K^{\text{ave-final}}$  versus total cars  $N$ : (a)  $M = 20$ ; (b)  $M = 100$ .

DCP algorithm. In particular,  $p_K^{\text{ave-final}}$  is given by

$$p_K^{\text{ave-final}} = \frac{1}{T} \sum_{t=1}^T p^{\text{final}}(t, K). \quad (30)$$

Note that when  $\mathbf{X}^{\text{cur}}(t, K)$  is feasible for all  $t \in \{1, \dots, T\}$ , then  $p_K^{\text{ave-final}} = p^{\text{ave}}(K)$  [compare with (29)].

Fig. 9 shows average objective value  $p_K^{\text{ave-final}}$  versus  $N$  for the cases  $M = 20$  [Fig. 9(a)] and  $M = 100$  [Fig. 9(b)]. Results show that DCP algorithm always outperforms the greedy method. Using subgradient iterations  $K = 500$  accounts for an increase in the performance gain compared with  $K = 300$ , though the gains are not substantial. When the setup is lightly loaded, proposed DCP performs very close to the optimal approach. For fixed  $M$ , increasing  $N$  results in increasing of the performance gap. For example, in the case of  $M = 20$  and  $N = 8$  with  $K = 300$  corresponds to a 3.94% performance deviation of DCP compared to the optimal CPLEX and  $N = 18$  corresponds to a 23.14% performance deviation, see Fig. 9(a). Moreover, in the case of  $M = 100$  and  $N = 50$  with  $K = 300$  corresponds to a 5.40% performance deviation of DCP compared to the optimal CPLEX and  $N = 95$  corresponds to a 259.20% performance

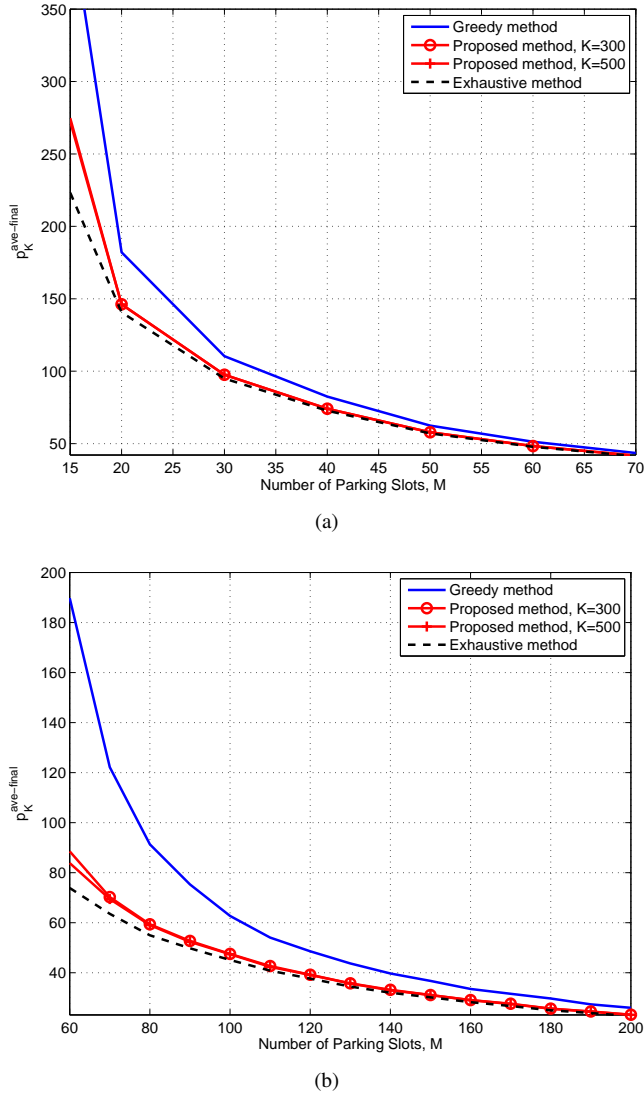


Fig. 10: Average objective value after the termination of DCP  $p_K^{\text{ave-final}}$  versus total parking slots  $M$ : (a)  $N = 10$ ; (b)  $N = 50$ .

deviation, Fig. 9(b). Such reductions in the performance gains are certainly expected, because there is a trade-off between the complexity of the algorithms and the performance loss. Results further show that the larger the parking slots  $M$ , the larger the performance deviation, especially in heavily loaded cases.

Fig. 10 shows average objective value  $p_K^{\text{ave-final}}$  versus  $M$  for the cases  $N = 10$  [Fig. 10(a)] and  $N = 50$  [Fig. 10(b)]. The observations are similar to those in Fig. 9. Results confirm that, our DCP algorithm performs very close to the optimal CPLEX method in lightly loaded cases, where  $N/M < 0.5$ , see Fig. 10(a) curves with  $M > 20$  and Fig. 10(b) curves with  $M > 100$ . However, there is a noticeable performance degradation in heavily loaded cases, see Fig. 10(a) curves with  $M < 20$  and Fig. 10(b) curves with  $M < 100$ . Moreover, the proposed method substantially outperforms the greedy method in all considered scenarios.

In order to provide a statistical description of the speed of the proposed algorithm, we consider empirical the cumulative distribution function (CDF) plots. Specifically, for each time slot  $t \in \{1, \dots, T\}$ , we store the total CPU time required for

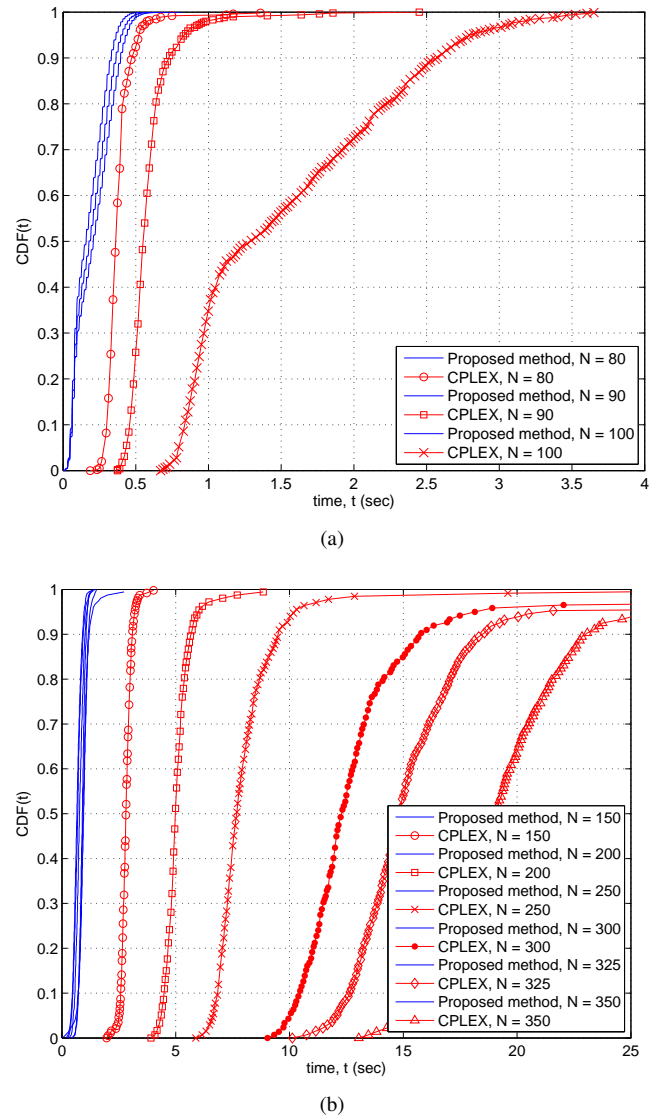


Fig. 11: CDF of time: (a)  $M = 100$ ; (b)  $M = 500$ .

DCP to find  $X^{\text{final}}(t, K)$ , where we use  $K = 300$ . Similarly, the total CPU time required to find the *optimal* value by using CPLEX is recorded. Figure 11 shows the empirical CDF plots of the time for  $N = 80, 90, 100$  with  $M = 100$  [Figure 11(a)] and for  $N = 150, 200, 250, 300, 325, 350$  with  $M = 500$  [Figure 11(b)]. In the case of DCP algorithm, the effects of changing the problem size by increasing  $N$  on the CDF plots are almost indistinguishable. However, in the case of optimal CPLEX method, there is a prominent increase in the time required to compute the optimal value. It should be emphasized that CPLEX finds the optimal assignment with a high penalty on time to do it, whereas DCP algorithm finds feasible assignment efficiently with a penalty on the optimality.

Figure 12 depicts the average time required by DCP algorithm, the optimal method, and the greedy method versus  $N$  for  $M = 100$  [Figure 12(a)] and for  $M = 500$  [Figure 12(b)]. Results show that the average time required by DCP to find possibly a suboptimal solution is not sensitive to the variation of  $N$  and is in the range  $0 - 1$  seconds. Moreover, they are comparable to the average time of simple greedy method. However, the average time required by the optimal method to

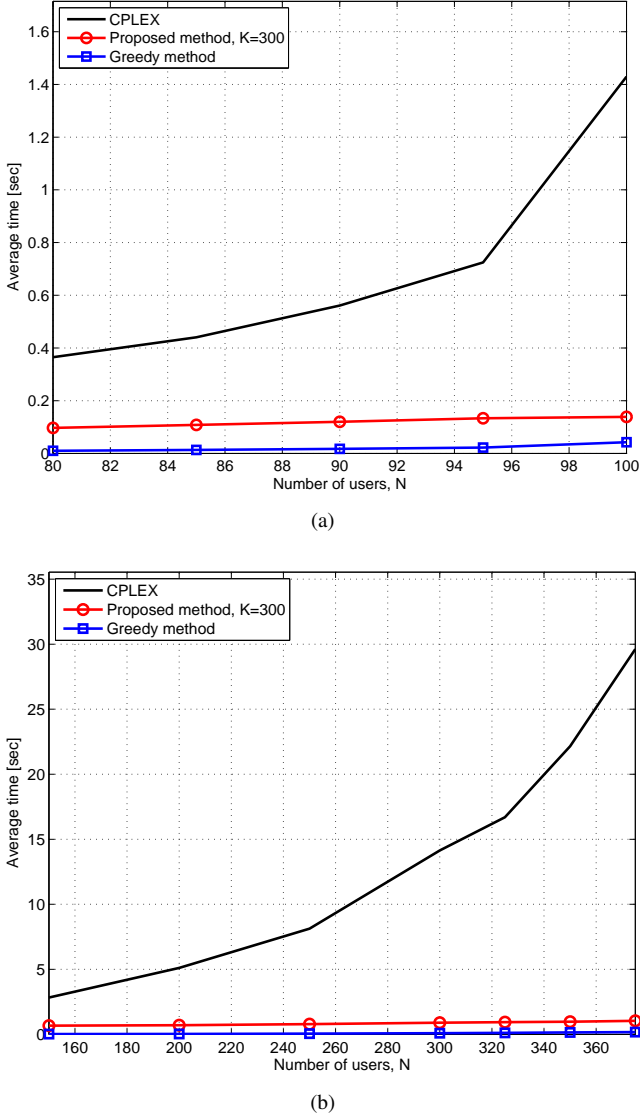


Fig. 12: Average CPU time: (a)  $M = 100$ ; (b)  $M = 500$ .

find the optimal solution grows approximately exponentially with  $N$ . This is certainly expected because problem (3) is combinatorial, and therefore the worst-case complexity of the optimal method grows exponentially with the problem size [23, § 1.4.2]. Thus, there is naturally a tradeoff between the optimality and the efficiency of the algorithms. The results together with those of Fig. 9 and 10 suggest that our proposed DCP algorithm yields a good tradeoff between the optimality and the efficiency, especially in lightly loaded cases. These properties are favorable for practical implementation.

## VII. CONCLUSIONS

In this paper, we considered the problem of car parking assignment. Unlike the existing greedy approaches, our problem formulation considered fairness among the scheduled cars in the sense that the global objective was to minimize the maximum distance from the parking slots to the intended destinations of the cars. A method based on Lagrange duality theory was proposed to address the nonconvex and combinatorial assignment problem. Even though we placed a stronger emphasis in the car parking slot assignment problem, our

formulation and the corresponding algorithm generally applies in fair agent-target assignment problems in other application domains as well. We highlighted appealing privacy properties of the proposed algorithm. In particular, we showed that the proposed method is privacy preserving in the sense that any car involved in the algorithm will not be able to discover the destination of any other car during the algorithm iterations. Unlike the optimal exponentially complex approaches, our proposed method is scalable. Roughly speaking, numerical results showed that for all considered cases, where the number of free parking slots are equal or higher than twice the scheduled cars, our proposed algorithm's performance is similar to that of the optimal method. Moreover, in all considered cases, the proposed algorithm outperformed the simple greedy approach. Therefore, the proposed algorithm yields a good trade-off between the implementation-level simplicity and the optimality.

## APPENDIX A

### THE EUCLIDEAN PROJECTION ONTO THE PROBABILITY SIMPLEX

In this appendix, we show how to project an arbitrary vector  $\mathbf{x} \in \mathbb{R}^N$  onto the probability simplex (14). This problem can formally be expressed as

$$\text{minimize } (1/2)\|\boldsymbol{\lambda} - \mathbf{x}\|_2^2 \quad (\text{A.1a})$$

$$\text{subject to } \mathbf{1}^T \boldsymbol{\lambda} = 1 \quad (\text{A.1b})$$

$$\boldsymbol{\lambda} \succeq \mathbf{0}, \quad (\text{A.1c})$$

where the variable is  $\boldsymbol{\lambda} \in \mathbb{R}^N$ . Here  $\mathbf{1}$  denotes the vector with all 1's,  $\mathbf{0}$  denotes the vector with all 0's, and  $\succeq$  denotes the component-wise inequality. The projection onto the probability simplex (14) is given by the solution  $\bar{\boldsymbol{\lambda}} = (\bar{\lambda}_1, \dots, \bar{\lambda}_N)$  of problem (A.1). We apply duality theory [23, § 5] to solve (A.1).

To do this, we first form the partial Lagrangian by dualizing the constraint (A.1b). Let  $\nu$  denote the associated multiplier. Then the partial Lagrangian  $L(\nu, \boldsymbol{\lambda})$  is given by

$$L(\nu, \boldsymbol{\lambda}) = (1/2)\|\boldsymbol{\lambda} - \mathbf{x}\|_2^2 + \nu(\mathbf{1}^T \boldsymbol{\lambda} - 1). \quad (\text{A.2})$$

The dual function  $h(\nu)$  is given by

$$h(\nu) = \inf_{\boldsymbol{\lambda} \succeq \mathbf{0}} L(\nu) = \inf_{\boldsymbol{\lambda} \succeq \mathbf{0}} (1/2)\|\boldsymbol{\lambda} - \mathbf{x}\|_2^2 + \nu(\mathbf{1}^T \boldsymbol{\lambda} - 1) \quad (\text{A.3a})$$

$$= \inf_{\boldsymbol{\lambda} \succeq \mathbf{0}} (1/2)\|\boldsymbol{\lambda} - (\mathbf{x} - \nu \mathbf{1})\|_2^2 + \nu(\mathbf{1}^T \mathbf{x} - 1) - (1/2)N\nu^2 \quad (\text{A.3b})$$

$$= (1/2) \sum_{i=1}^N (\min\{0, x_i - \nu\})^2 + \nu(\mathbf{1}^T \mathbf{x} - 1) - (1/2)N\nu^2, \quad (\text{A.3c})$$

where the equality (A.3b) follows from straightforward mathematical manipulations and (A.3c) follows from that  $\lambda_i = 0$  if  $(x_i - \nu) \leq 0$  and  $\lambda_i = (x_i - \nu)$  if  $(x_i - \nu) > 0$ . The dual optimal value  $\nu^*$  is given by  $\nu^* = \arg \max_{\nu} h(\nu)$  [23, § 5.2]

Note that the dual function  $h(\nu)$  is a scalar valued function. Moreover, a subgradient  $r(\nu)$  of  $h(\nu)$  at  $\nu$  can be analytically computed as

$$r(\nu) = - \sum_{i=1}^N (\min\{0, x_i - \nu\}) + \mathbf{1}^T \mathbf{x} - 1 - N\nu. \quad (\text{A.4})$$



Note that the dual function is always concave. Thus the sign of its subgradients changes as we pass through the maximum point (i.e.,  $\nu^*$ ) of the dual function, which allows us to use a bisection search method to find  $\nu^*$  as follows.

---

**Algorithm:** COMPUTATION OF  $\nu^*$

- 1) Given the accuracy level  $\epsilon > 0$ ,  $\nu^{\min}$  and  $\nu^{\max}$  such that  $r(\nu^{\min}) > 0$  and  $r(\nu^{\max}) < 0$ .
  - 2) If  $\nu^{\max} - \nu^{\min} < \epsilon$ , return  $\nu^* = (\nu^{\max} + \nu^{\min})/2$  and STOP.
  - 3) Set  $\nu = (\nu^{\max} + \nu^{\min})/2$ .
  - 4) If  $r(\nu) \geq 0$ , set  $\nu^{\min} = \nu$ . Otherwise, set  $\nu^{\max} = \nu$ . Go to step 2.
- 

Because strong duality holds for problem (A.1) [23, § 5.2.3], we can recover its solution  $\bar{\lambda} = (\bar{\lambda}_1, \dots, \bar{\lambda}_N)$  as

$$\bar{\lambda}_i = \max\{0, x_i - \nu^*\}, \quad i = 1, \dots, N. \quad (\text{A.5})$$

## REFERENCES

- [1] J-P Rodrigue, C. Comtois, and B. Slack, *The Geography of Transport Systems*, New York: Routledge, 2nd edition, 2013, [Online]. Available: <http://people.hofstra.edu/geotrans>.
- [2] D. C. Shoup, "Cruising for parking," *Transport Policy Special Issue on Parking*, vol. 13, no. 6, pp. 479–486, 2006.
- [3] B. Chen and H.H. Cheng, "A review of the applications of agent technology in traffic and transportation systems," *IEEE Trans. Intelligent Transportation Sys.*, vol. 11, no. 2, pp. 485–497, 2010.
- [4] H. Jianming, M. Qiang, W. Qi, Z. Jiajie, and Z. Yi, "Traffic congestion identification based on image processing," *IET Intelligent Transport Sys.*, vol. 6, no. 2, pp. 153–160, 2012.
- [5] W-H Lee, S-S Tseng, J-L Shieh, and H-H Chen, "Discovering traffic bottlenecks in an urban network by spatiotemporal data mining on location-based services," *IEEE Trans. Intelligent Transportation Sys.*, vol. 12, no. 4, pp. 1047–1056, 2011.
- [6] G. Dimitrakopoulos, P. Demestichas, and V. Koutra, "Intelligent management functionality for improving transportation efficiency by means of the car pooling concept," *IEEE Trans. Intelligent Transportation Sys.*, vol. 13, no. 2, pp. 424–436, 2012.
- [7] K. Chiew and Q. Shaowen, "Scheduling and routing of AMOs in an intelligent transport system," *IEEE Trans. Intelligent Transportation Sys.*, vol. 10, no. 3, pp. 547–552, 2009.
- [8] M. Ergen, S. Coleri, Y. Shon, M. Ozalp, and A. Long, "EzPARK," [Online]. Available: <http://www.eecs.berkeley.edu/ergen/EZPARK/index.htm>, 2003.
- [9] M. Grossglauser and M. Piorkowski, "SmartPark," [Online]. Available: <http://smartpark.epfl.ch/index.html>, 2006.
- [10] M. Piorkowski, M. Grossglauser, and A. Papaioannou, "Mobile user navigation supported by WSN: Full-fledge demo of the smartpark system," in *Mobile Ad Hoc Networking and Computing*, Florence, Italy, May 22–25 2006.
- [11] M. Piorkowski, M. Grossglauser, and A. Papaioannou, "SmartPark: High-mobility application supported by wireless sensor and actuator network," in *Mobile Information & Communication Systems (MICS)*, Zurich, Switzerland, Oct. 17–19 2006.
- [12] V.W.S. Tang, Y. Zheng, and J. Cao, "An intelligent car park management system based on wireless sensor networks," in *Pervasive Computing and Applications, 2006 1st International Symposium on*, 2006, pp. 65–70.
- [13] Y-Z. Bi, L-M. Sun, H-S. Zhu, T-X. Yan, and Z-J. Luo, "A parking management system based on wireless sensor network," in *ACTA AUTOMATICA SINICA*, 2011, vol. 32, pp. 38–45.
- [14] J. Chinrungrueng, U. Sunantachakul, and S. Triamlumlerd, "Smart parking: An application of optical wireless sensor network," in *Applications and the Internet Workshops, 2007. SAINT Workshops 2007. International Symposium on*, 2007, pp. 66–66.
- [15] L. Rongxing, X. Lin, H. Zhu, and X. Shen, "Spark: A new vanet-based smart parking scheme for large parking lots," in *INFOCOM 2009, IEEE*, 2009, pp. 1413–1421.
- [16] L. Rongxing, X. Lin, H. Zhu, and X. Shen, "An intelligent secure and privacy-preserving parking scheme through vehicular communications," *IEEE Trans. Veh. Tech.*, vol. 59, no. 6, pp. 2772–2785, 2010.
- [17] R. Souissi, O. Cheikhrouhou, I. Kammoun, and M. Abid, "A parking management system using wireless sensor networks," in *Microelectronics (ICM), 2011 International Conference on*, 2011, pp. 1–7.
- [18] ParkingCarma, "Smartparking information network (SPIN)," [Online]. Available: <http://www.parkingcarma.com>, 2007.
- [19] Streetline Networks Inc., "Parker mobile," [Online]. Available: <http://www.streetline.com/find-parking/>.
- [20] VehicleSense Inc., "Street parking information network (SPIN)," [Online]. Available: [http://www.vehiclesense.com/vs\\_solutions.html](http://www.vehiclesense.com/vs_solutions.html).
- [21] San Francisco Municipal Transportation Agency (SFMTA), "SFpark," [Online]. Available: <http://sfpark.org/>.
- [22] B. Radunović and J-Y Le Boudec, "A unified framework for max-min and min-max fairness with applications," *IEEE/ACM Trans. Netw.*, vol. 15, no. 5, pp. 1073–1083, oct 2007.
- [23] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.
- [24] S. Boyd, "Primal and dual decomposition," [Online]. Available: [http://www.stanford.edu/class/ee364b/lectures/decomposition\\_slides.pdf](http://www.stanford.edu/class/ee364b/lectures/decomposition_slides.pdf), 2007.
- [25] S. Boyd, "Subgradient methods," [Online]. Available: [http://www.stanford.edu/class/ee364b/lectures/subgrad\\_method\\_slides.pdf](http://www.stanford.edu/class/ee364b/lectures/subgrad_method_slides.pdf), 2007.
- [26] R. Horst, P. Pardalos, and N. Thoai, "Introduction to global optimization," vol. 48, 2000.
- [27] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, MA, 2nd edition, 1999.
- [28] O. Goldreich, *The Foundations of Cryptography*, vol. 2, Cambridge University Press, Cambridge, UK, 2004.
- [29] "IBM ILOG CPLEX Optimizer. [online]. Available: <http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/>."



**Elisabetta Alfonsetti** received the Master degree in Computer Science from University of L'aquila, Italy, in 2012. Then she worked as research engineer in the Automatic Control Lab, Electrical Engineering Department and ACCESS Linnaeus Center, KTH Royal Institute of Technology, Stockholm, Sweden, focusing on optimization techniques and privacy preserving issue. She is currently with the TerraSwarm Lab, Electrical Engineering Department, UC Berkeley, California. Her research interests include application of Contract-based design paradigm for the

design of complex systems.



**Pradeep Chathuranga Weeraddana** (S'08, M'11) received the M.Eng degree Telecommunication from School of Engineering and Technology, Asian Institute of Technology, Thailand in 2007 and the Ph.D. degree from University of Oulu, Finland, in 2011. He is currently working as postdoctoral researcher in Automatic Control Lab, Electrical Engineering Department and ACCESS Linnaeus Center, KTH Royal Institute of Technology, Stockholm, Sweden. His research interests include application of optimization techniques in various application domains, such as signal processing, wireless communications, smart grids, privacy, and security.





**Carlo Fischione** (M'05) is a tenured associate professor at KTH Royal Institute of Technology, Electrical Engineering, Stockholm, Sweden. His research interests include optimization and parallel computation with applications to wireless sensor networks, networked control systems, and wireless networks. He received the best paper award from the IEEE Transactions on Industrial Informatics of 2007. He is a member of the IEEE and SIAM.